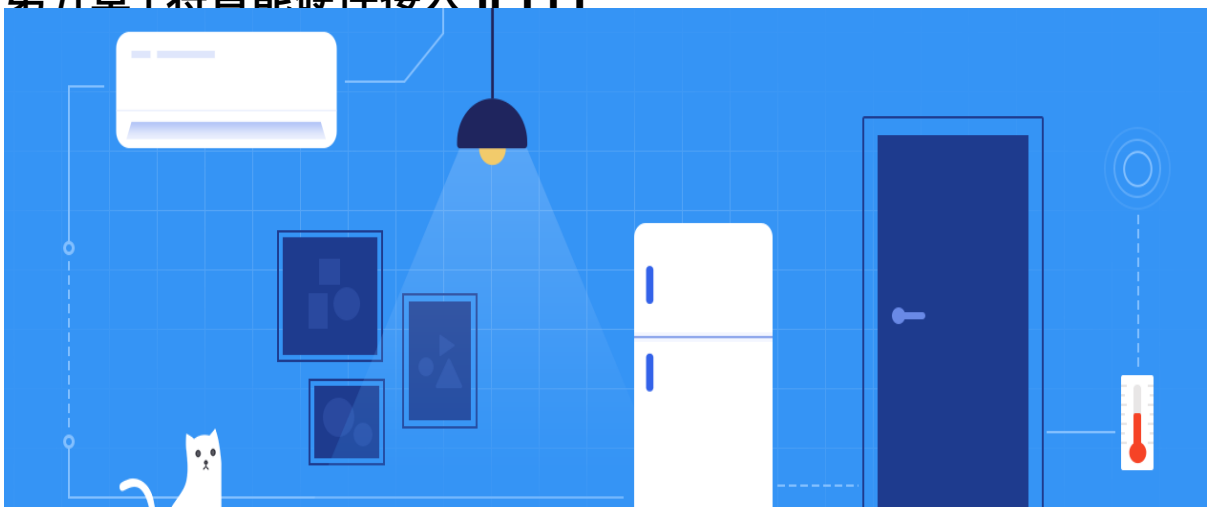


王掌柜

05月13日

第五章 | 将智能硬件接入 IFTTT

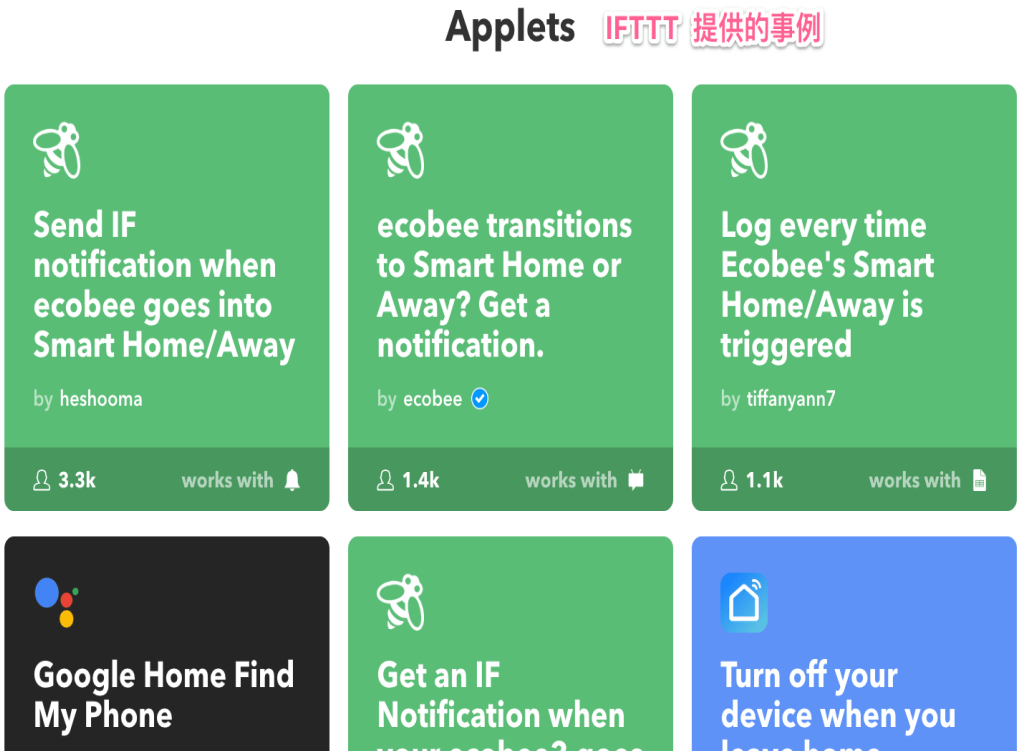
| 本文为付费栏目文章，您已订阅，可阅读全文 |

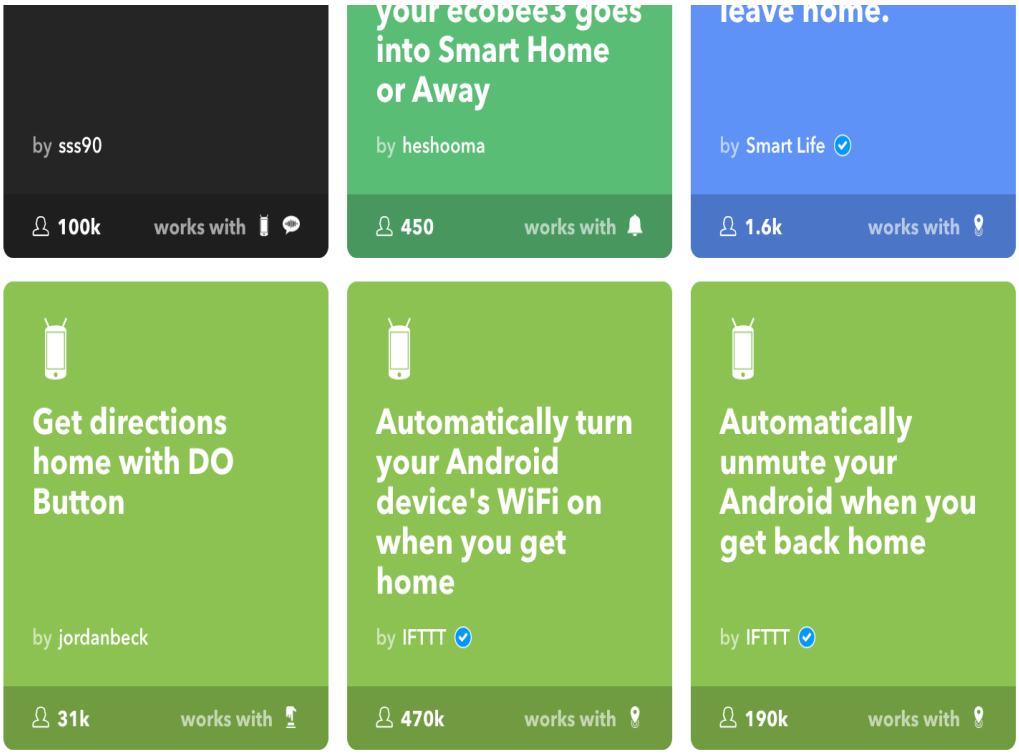
智能硬件发展到今天已经是可以普及到千家万户的事情了。整个智能硬件领域如火如荼，各大服务商都在为智能硬件的普及铺路，从 Apple、Google、亚马逊、Philips 等国际品牌，到阿里、腾讯、小米等国内品牌，都在智能硬件领域有所建树。在做硬件的同时，各家也陆续推出了自己的「智能硬件平台」，通过平台将自家的智能硬件链接在一起完成复杂的自动化任务。例如 Apple 的 HomeKit，Google 的 Goolge Home，Philips 的 Hue，小米的「米家」。

既然各家都提供了「智能硬件平台」，那么为什么还要研究「将智能硬件接入到 IFTTT」？

1. IFTTT 作为最早接入智能硬件的智能服务运营商，已经积累了大量的智能硬件服务：

IFTTT 支持的智能硬件服务





IFTTT 支持的智能硬件实例

2. IFTTT 通过多年的积累，已经支持了大量的互联网服务，智能硬件接入 IFTTT 就相当于接入了 IFTTT 的巨大的资源库，随时可以派生出各种有趣的玩法。

接下来，我们以小米的「米家」和 Philips 的 Hue 接入到 IFTTT 为例，了解 IFTTT 的更多进阶玩法。

01. IFTTT 和 小米智能硬件的进阶玩法

智能硬件发展到今天已经非常普及了，整个领域做平台的做平台，做硬件的做硬件，一片欣欣向荣。在这其中，小米公司凭借「米家」出色的性价比，在国内外智能硬件领域都有着不错的口碑。

米家提供了众多的智能硬件，它们都要连接到「米家多功能网关」上，由网关负责调度和处理个智能硬件的业务。

通过米家 app，我们可以管理大多数的智能硬件，在客户端里我们可以查看硬件的执行状态，查看各种数据，甚至是创造「自动化任务」。

我们看到「米家」自身功能就很丰富，无论是远程控制，还是自动化流程，都完成的非常好，如果你只是玩转米家自身的硬件，米家平台就足够了，但是如果想**接入更多国外的服务**，就必须借助 IFTTT 了。

明白了 IFTTT 对米家智能硬件的作用，我们正式开启米家的 IFTTT 接入之旅。

Home Assistant

「米家」是无法直接接入到 IFTTT 的，这里我们需要借助 Home Assistant 平台，Home Assistant 是一个第三方智能家居服务平台。Home Assistant 支持米家设备的接入，同时它又可以接入 IFTTT，从而帮我们将「米家」链接到 IFTTT。

如果要跟着步骤操作，需要开启「米家智能网关」，并且至少拥有一个米家智能硬件，这里推荐「米家无线开关」。

Home Assistant 需要安装在一台 24 小时开机的设备上，我用的是 Mac mini，因此本文中会以 Mac mini 为例进行操作。

24 小时开机的设备还有个更普遍的选择是树莓派。少数派有一篇比较详细的文章——《入门智能家居，从米家到 HomeKit（一）》，其中介绍了用树莓派安装 HA 的方法，操作也不复杂。如果你用的是树莓派的话，可以参考该 《入门

智能家居，从米家到 HomeKit（一）》 安装 Home Assistant 再继续看「二、配置 Home Assistant 接入小米设备」以下的教程。

一、安装 Home Assistant

Mac OS 上可以通过「终端（Terminal）」进行命令行[2] 操作，最多只需要 3 行命令就可以安装完成。

1. 安装 brew（如果已经安装请忽略）

   brew 是 macOS 平台上优秀的「包管理器」，支持通过命令行方式安装绝大多数服务和软件，它的安装方式如下，在命令行中输入以下指令「回车」即可：

   ```
   /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercon
   tent.com/Homebrew/install/master/install)"
   ```

2. 安装 pip3 （如果已经安装请忽略）

   pip 是一个现代的，通用的 Python 包管理工具。提供了对 Python 包的查找、下载、安装、卸载的功能。安装方式如下，在命令行中输入以下指令「回车」即可（python3 包含了 pip3）：

   ```
   brew install python3
   ```

3. 安装 Home Assistant，在命令行中输入以下指令「回车」即可（此命令树莓派通用）：

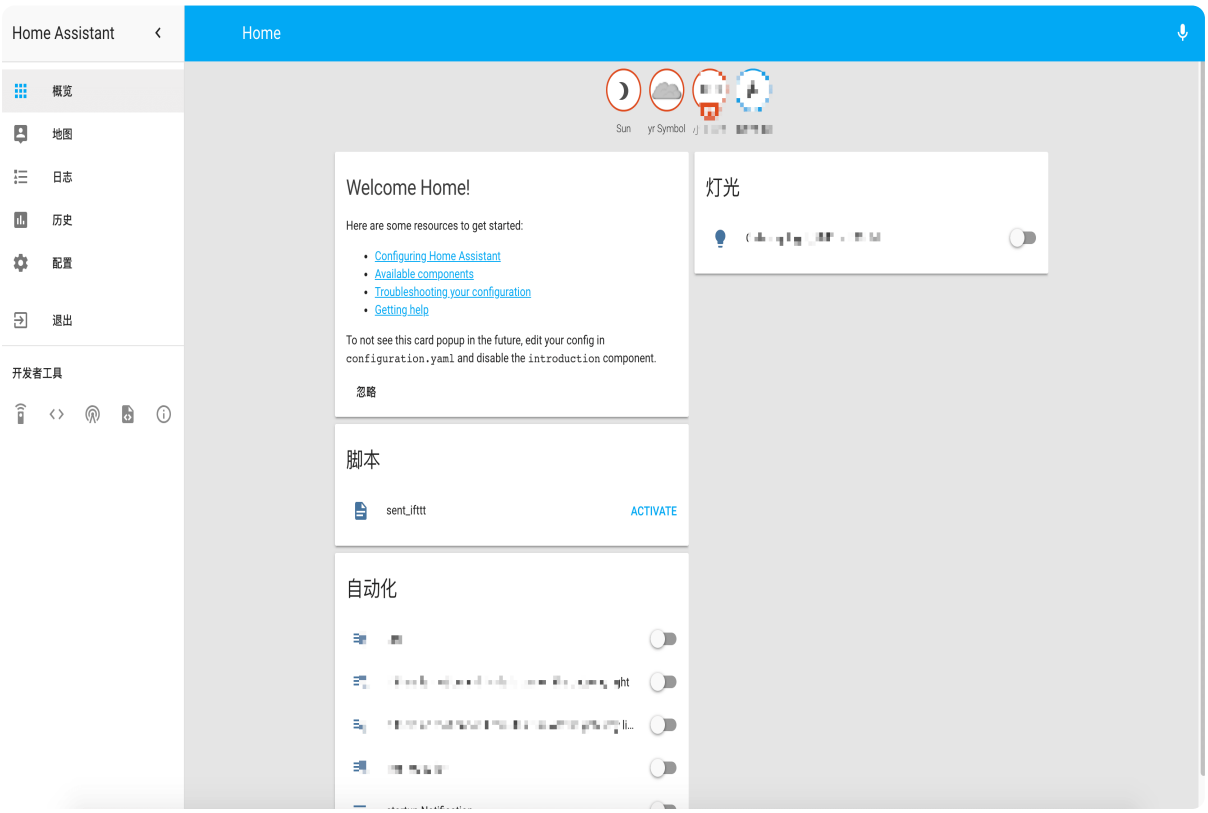
   ```
   pip3 install homeassistant
   ```

如果一切顺利，你就安装好 Home Assistant 了，接下来我们启动它，同样在命

令行中输入以下指令「回车」即可：

```
hass --open-ui
```

启动成功后，它会自动打开浏览器访问 Web 页面：



Home Assistant 的页面

Tips: 如果浏览器中的 IP 地址不是 `127.0.0.1` 可能会遇到浏览器打不开的状况，此时可以将浏览器中的 IP 地址改成 `127.0.0.1` 再刷新页面即可。

二、配置 Home Assistant 接入小米设备

Tips：接下来的部分操作需要用到 VI[3] 命令，推荐阅读《vi 编辑器的简单介绍和使用》了解基本用法。

首先**修改配置文件**，在命令行中输入如下指令「回车」即可：

```
```

```
vi ~/.homeassistant/configuration.yaml
```

接下来我们需要将以下内容插入到配置文件的最后：

```
## 修改配置识别米家控制器
xiaomi_aqara:
  gateways:
    - mac: 781[redacted]9D
      key: 5B0B[redacted]
```

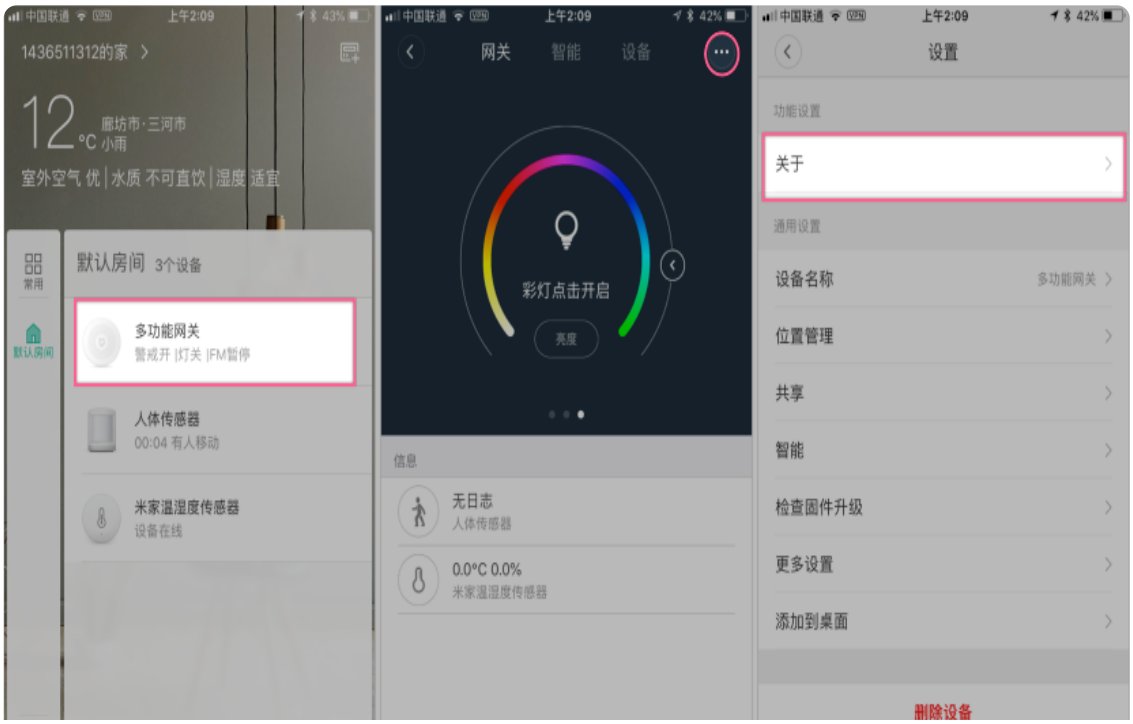
操作方法看视频：

主要操作指令（注意输入法要一直在英文状态下）：

1. 光标默认停在第一行第一个字符上。
2. 按下快捷键 `⇧Shift + G` 光标会跳转到最后一行。
3. 如果最后一行有内容，按下快捷键 `⇧Shift + 4` 光标跳转到最后一个字符。
4. 按下 `A` 键，然后回车，此时处于编辑状态。
5. 输入刚才图中内容，注意图中的对齐方式。
6. 输入完毕后，按下 `Esc` 键，退出编辑状态。
7. 按下 `:`，进入指令输入模式。
8. 在 `:` 后输入小写字母 `w` 后，配置文件修改被保存。
9. 最后再次按下 `:` 输入小写字母 `q`，退出编辑器，文件修改成功。

上图中的 `mac` 和 `key` 可以通过以下方式找到：

1. 打开「米家 app」，选择智能网关设备，点击右上角 `···` 符号，进入后点击「关于」选项。

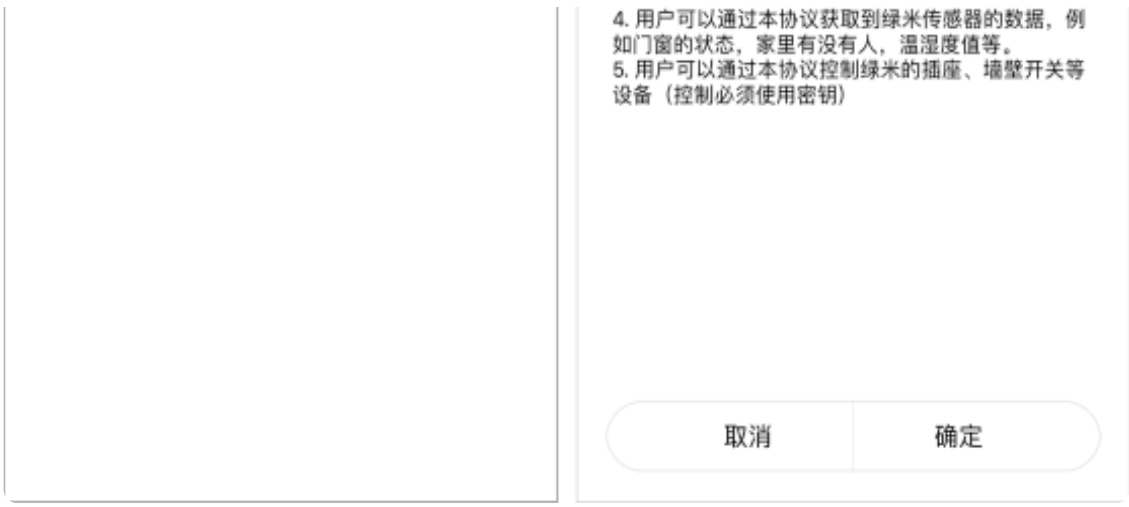


2. 在「左图」空白处「快速点击」，稍后界面会变成「右图」样子。

3. 获取 `mac` 和 `key` 的值：

智能家庭套装论坛
套装版本:4.8.0, 主版本:4.7.4
局域网通信协议
网关信息
玩法教程
网关ID: 78898167
Zigbee通道: 26
网关信息:
{ otu_stat=(83, 109, 497, 0, 497,
12), mac=78:11:[redacted]:0F:[redacted], life=16669,
nw_ver=MW300, ott_stat=(0, 0, 0, 0),
wifi_fw_ver=SD878x-14.76.36.p84-702.1.0-WM,
fw_ver=1.4.1_154,
token=42fe479cd50e58037885293f72a34acb,
netif={ localip=192.168.199.123,
mask=255.255.255.0, gw=192.168.199.1,
gw_mac=D4:EE:07:3A:F7:CA}, cfg_time=0,
mcu_fw_ver=0143, ap={ rssi=-61,
bssid=D4:EE:07:3A:F7:CA, ssid=HiWiFi_3AF7CA},
mmfree=170984, ot=otu,
model=lumi.gateway.v3}
子设备信息:
{ Model: lumi.sensor_motion.v2,did: lumi.
158d0002245c9d,name: 人体传感器 }{ Model:
lumi.sensor_ht.v1,did: lumi.158d0001e723bc,name:
米家温湿度传感器 }
加密
复制

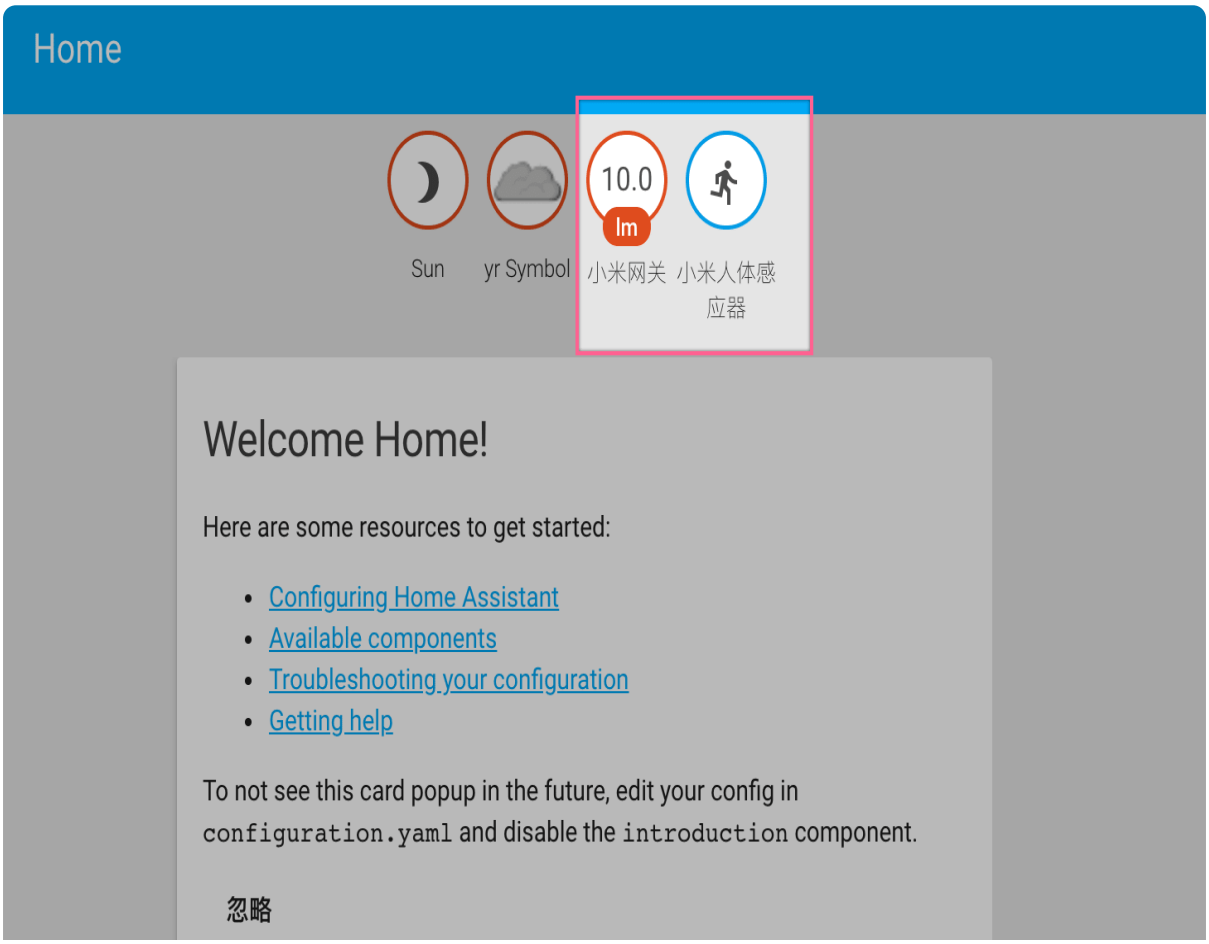
中国联通 上午2:09 42%
关于
智能家庭套装论坛
套装版本:4.8.0, 主版本:4.7.4
局域网通信协议
网关信息
玩法教程
中国联通 上午2:53 38%
局域网通信协议
密码: 5B[redacted]
多功能网关局域网通讯协议
1. 开放本协议的目的是为了让不同厂家的设备能够互联互通，共同组建物联网。
2. 用户可以把绿米的设备集成到homekit或其它现有系统中。
3. 本协议只在局域网中通讯，使用的协议或机制主要有：udp协议，组播，AES加解密。



Tips：注意 `mac` 地址写入配置文件时不需要带着冒号 `:`。

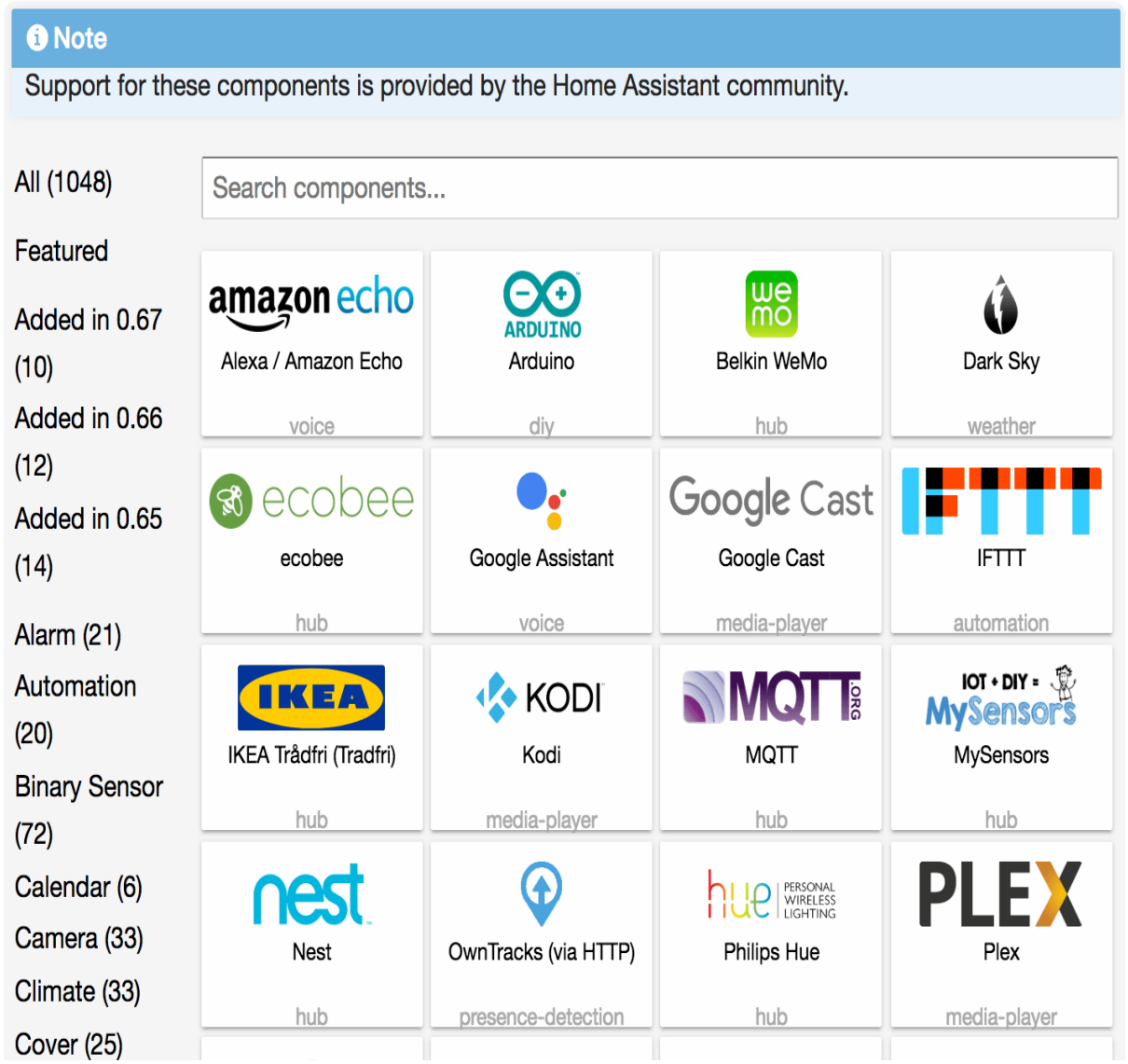
然后**重启服务**：可以通过 Web 端进行重启，点击首页左侧「配置」菜单，进入页面后点击「通用」菜单，再进入页面找到「服务管理」，点击「重启服务」即可。重启成功后，会重新打开管理页面。

重启后我们可以看到在 Web 端已经显示出小米的设备了如图：



三、Home Assistant 开启 IFTTT 组件

Home Assistant 有着丰富的组件，这也是它的最大特点，点击官网可以查询它支持的所有组件。



我们可以看到官方是支持 IFTTT 的，配置也起来很简单：

1. 修改配置文件，在命令行中输入如下指令「回车」即可。

```
vi ~/.homeassistant/configuration.yaml
```

2. 在文件末尾追加如下数据，（注意每一行的空格要和下图一致）：

```
ifttt:
  key:nwC2t_Tk1gob1Ac-5gIGxpX
```

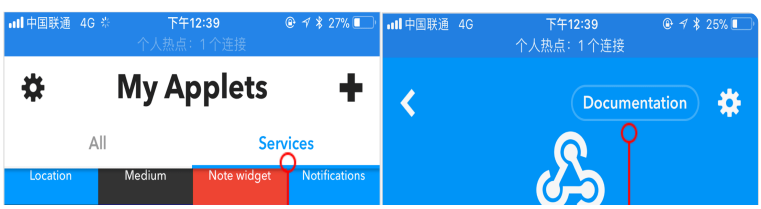
操作方法看视频：

主要操作指令（注意输入法要一直在英文状态下）：

1. 光标默认停在第一行第一个字符上。
2. 按下快捷键 `⇧Shift + G` 光标会跳转到最后一行。
3. 如果最后一行有内容，按下快捷键 `⇧Shift + 4` 光标跳转到最后一个字符。
4. 按下 `A` 键，然后回车，此时处于编辑状态。
5. 输入刚才图中内容，注意图中的对齐方式。
6. 输入完毕后，按下 `Esc` 键，退出编辑状态。
7. 按下 `:`，进入指令输入模式。
8. 在 `:` 后输入小写字母 `w` 后，配置文件修改被保存。
9. 最后再次按下 `:` 输入小写字母 `q`，退出编辑器，文件修改成功。

我们看到，这里也需要一个 key ， 这个 key 如何获得呢?

在《进阶篇》中我们提到过，IFTTT 给每个人分配了一个 `key` ，这里也是用 webhooks 的 `key` ，大家可以点此查看自己的 key ， 也可以在 IFTTT 客户端中查看：

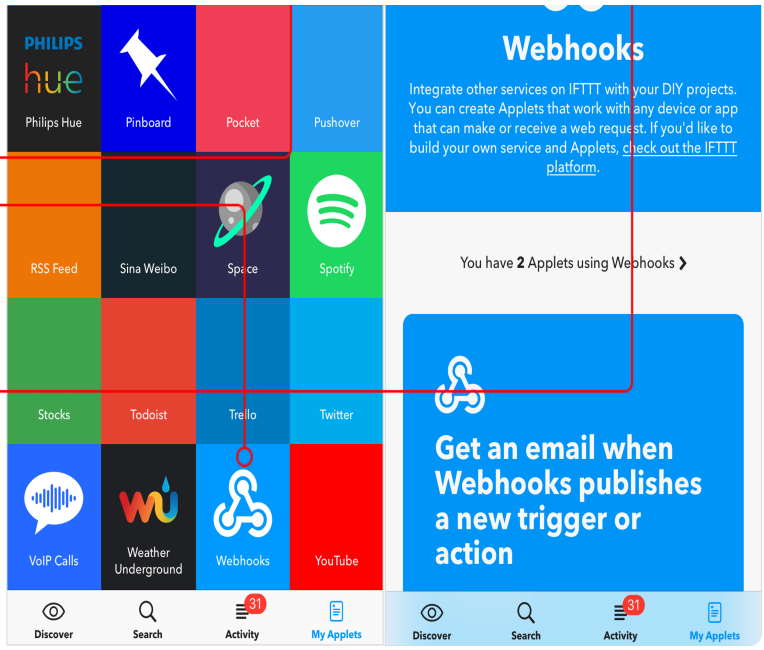


查看 Webhooks：

1. 切换到 Service 标签

2. 点击查看 Webhooks

3. 点击查看文档

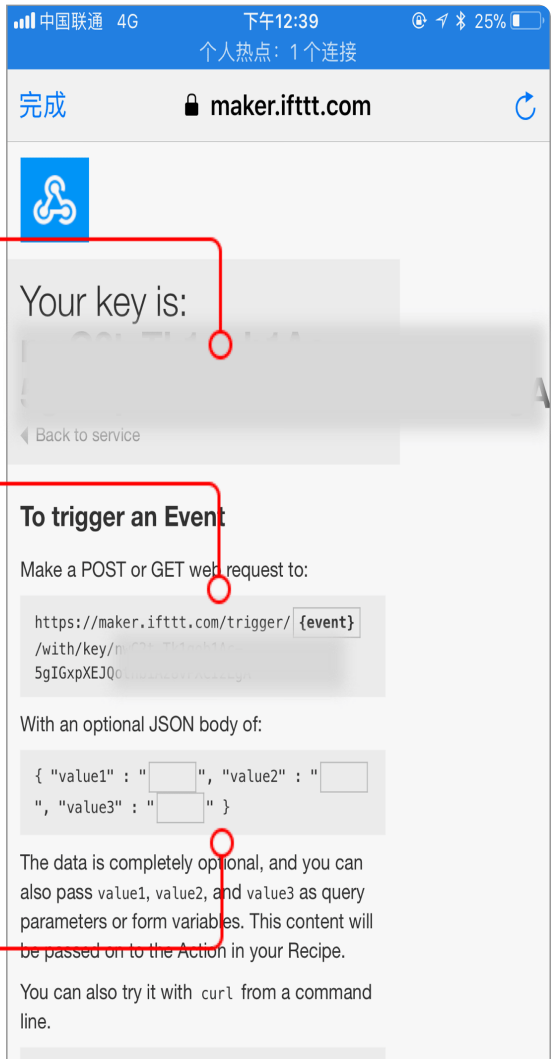


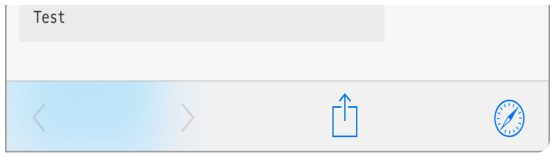
设置 Webhooks：

1. IFTTT 分配给你的 Key

2. 可以通过 POST 或者 GET 请求以下地址
注意{event}是「事件名」，用来区分各种 Webhooks

3. JSON 格式请求参数
目前 IFTTT 支持最多三个参数

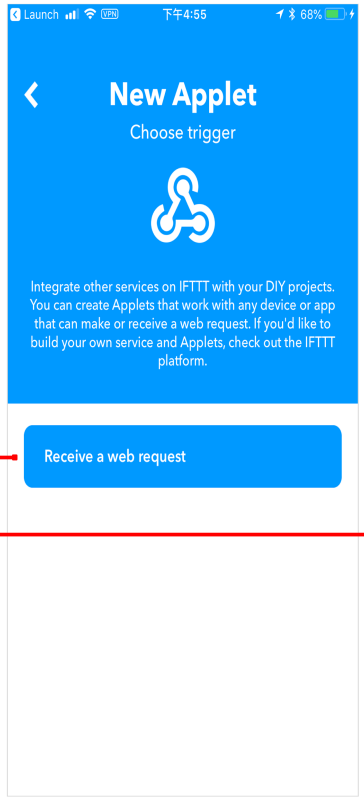

中国联通 4G
下午12:39
25%
个人热点：1 个连接
完成
maker.ifttt.com
Your key is:
Back to service
To trigger an Event
Make a POST or GET web request to:
https://maker.ifttt.com/trigger/{event}
/with/key/
With an optional JSON body of:
{ "value1" : "", "value2" : "", "value3" : "" }
The data is completely optional, and you can also pass value1, value2, and value3 as query parameters or form variables. This content will be passed on to the Action in your Recipe.
You can also try it with curl from a command line.

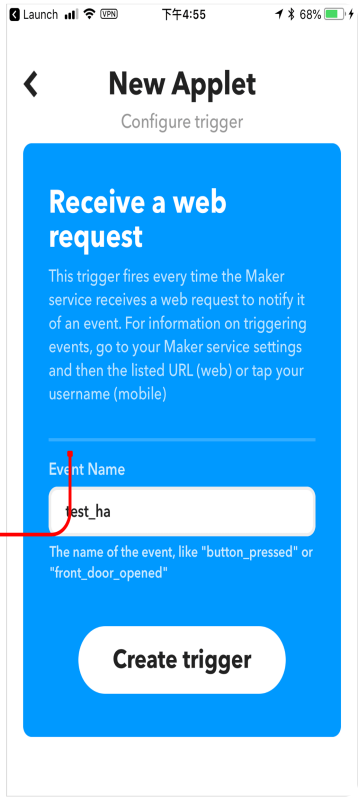


最后重启 Home Assistant，就完成了 IFTTT 组件的安装。

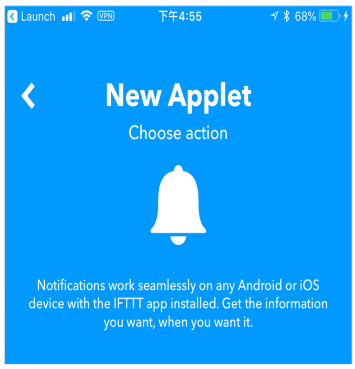
测试 IFTTT 组件是否成功：

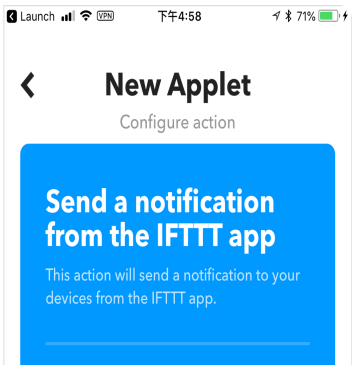
1. 创建测试用的 webhooks，关键步骤如下。

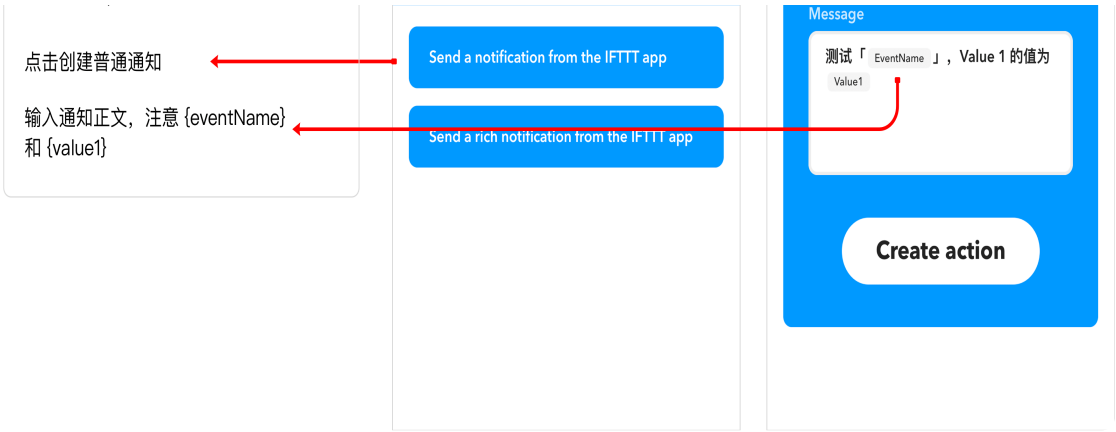




设置动作，发送通知：



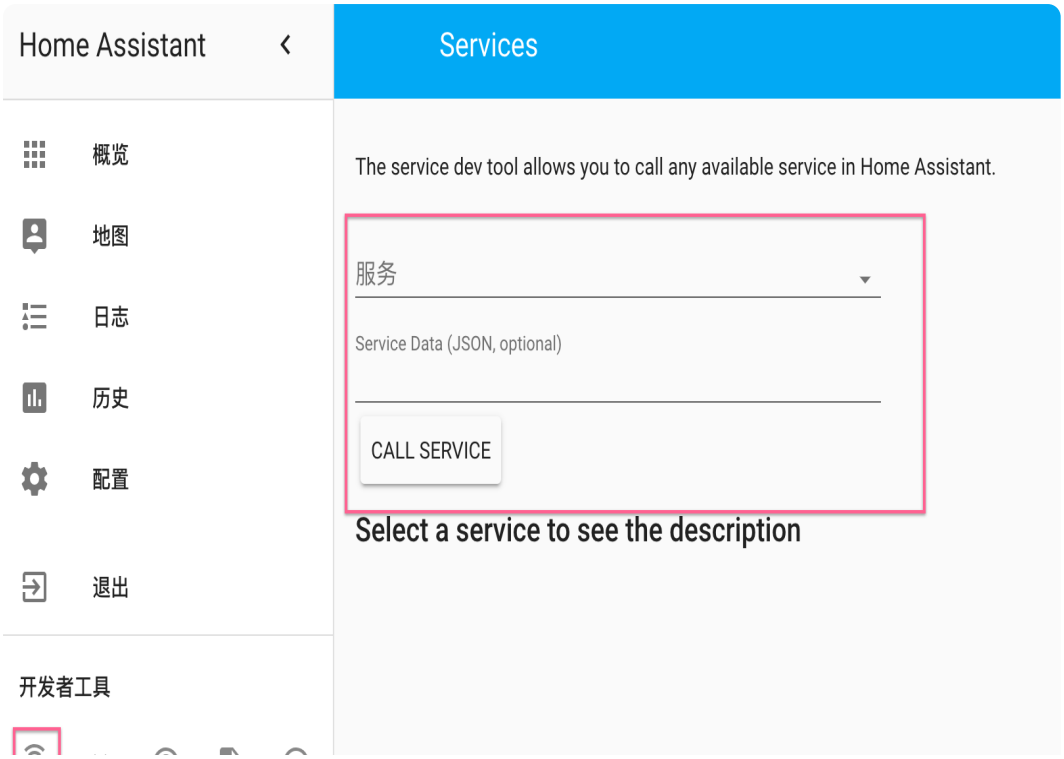


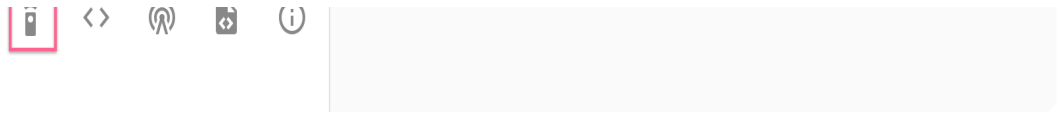


Tips: 注意设置「发送通知」动作时，用到了两个变量 `{EventName}` 和 `{Value1}`。分别代表被调用的 webhooks 的名称（上图中填写的 `test_ha`），和调用方传过来的参数名为 `Value1` 对应的值。

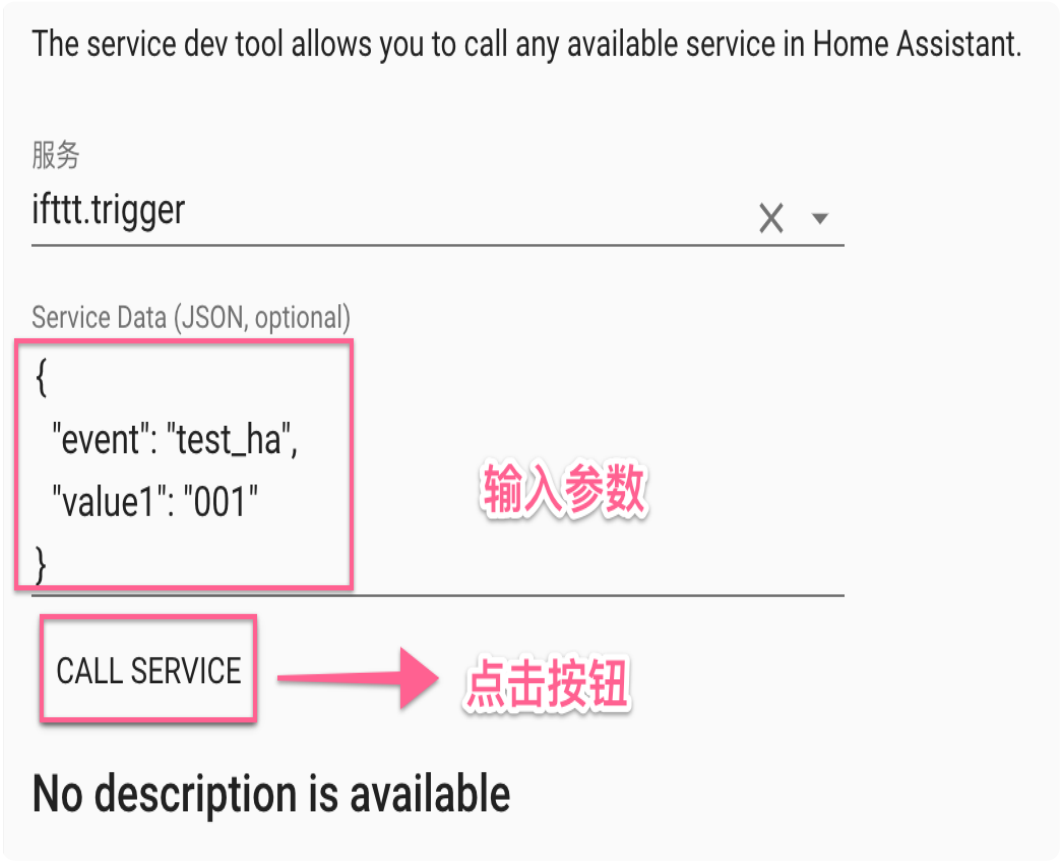
2. 在 Home Assistant 上测试 IFTTT 组件。

    1. 访问 web 页面，点击「左侧边栏」最左侧的图标，打开 Service 页面，如图：





2. 选择服务为 `ifttt.trigger`，设置 Service Data 为 `{"event": "test_ha","value1": "001"}`，并点击 Call SERVICE 按钮，如图：



点击按钮后，Home Assistant 会调用 IFTTT 中名称为 `test_ha` 的 webhooks 服务，并且传递参数 `Value1=001` 。

3. 调用成功，手机收到推送通知：

至此，通过以上三步，我们已经成功的将「米家」和 IFTTT 接入到 Home Assistant 中。接下来我们尝试一下「米家」+ IFTTT 的智能任务。

事例 1：如果「米家无线开关」被按住，那么让我的手机发出警报声

当我们在家的时候很容易把手机随手一扔，等你想找的时候找不到了，此时我们可以通过长按「米家无线开关」，让手机发出警报声，那么就很容易找到手机了。

按惯例，先分析一下这个事例：

按惯例，先分析一下这个事例：

- Trigger：如果按住「米家无线开关」的按钮。
- Action：手机发出警报声。

通过分析我们知道，完成这个任务我们需要分成两部分：

1. Home Assistant 监控到「米家无线开关」被按住。
2. Home Assistant 调用 IFTTT 的 webhooks 让手机发出警报声。

第一步， Home Assistant 监控到「米家无线开关」按住事件

Home Assistant 为我们提供了 Automation （自动化）服务，通过简单的配置就可以创建出「自动化任务」，我们先请检查配置文件中是否开启了 Automation，通过如下「命令」检查：

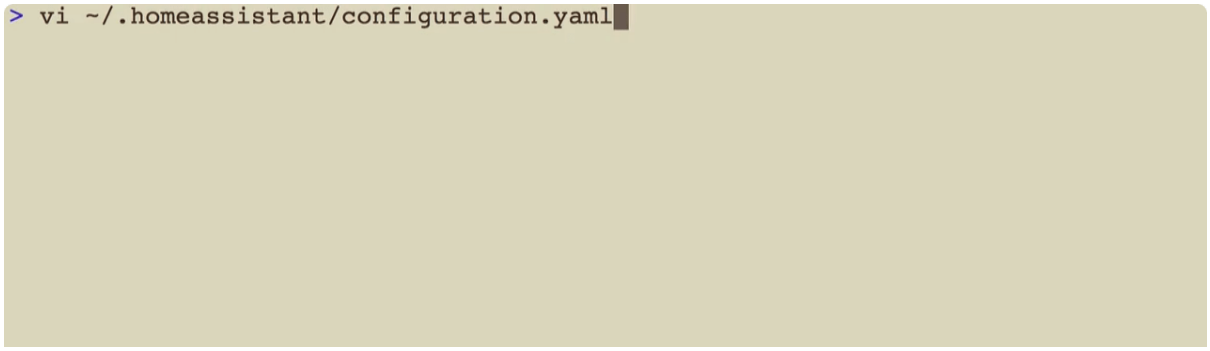
```
vi ~/.homeassistant/configuration.yaml
```

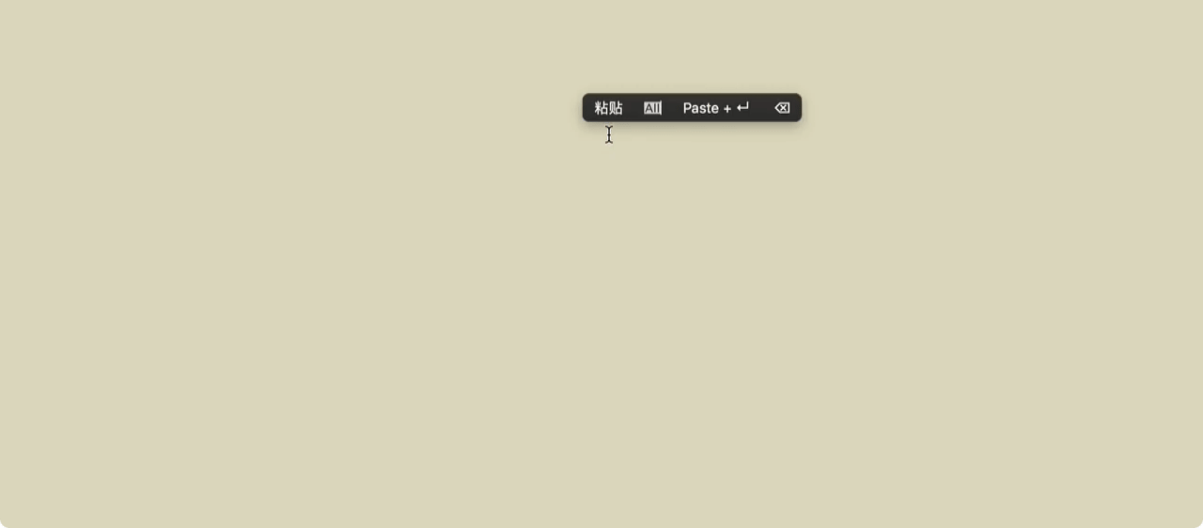
找到下边这行数据：

```
automation: !include automations.yaml
```

如果最前边有 `#` 请删除 `#` 并重启 Home Assistant。

操作方法看视频：



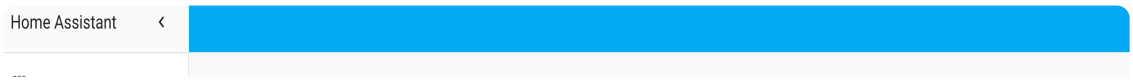


主要操作指令（注意输入法要一直在英文状态下）：

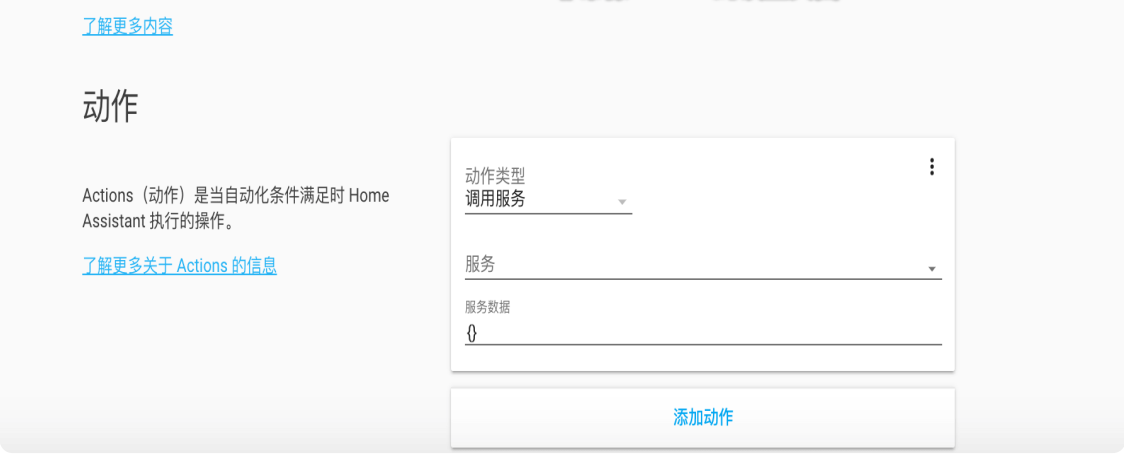
1. 光标默认停在第一行第一个字符上。
2. 按方向键 `↓`，直到光标移动到 `#automation: !include automations.yaml` 所在的行。
3. 按 `I` 键进入编辑状态。
4. 删除 `#`。
5. 按下 `Esc` 键，退出编辑状态。
6. 按下 `:`，进入指令输入模式。
7. 在 `:` 后输入小写字母 `w` 后，配置文件修改被保存。
8. 最后再次按下 `:` 输入小写字母 `q`，退出编辑器，文件修改成功。

确定没有问题，我们开始创建「第一个自动化任务」：

1. 通过 Home Assistant 创建自动化任务。点击页面左侧边栏的「配置」按钮，然后再点击「自动化」按钮进入自定义页面。

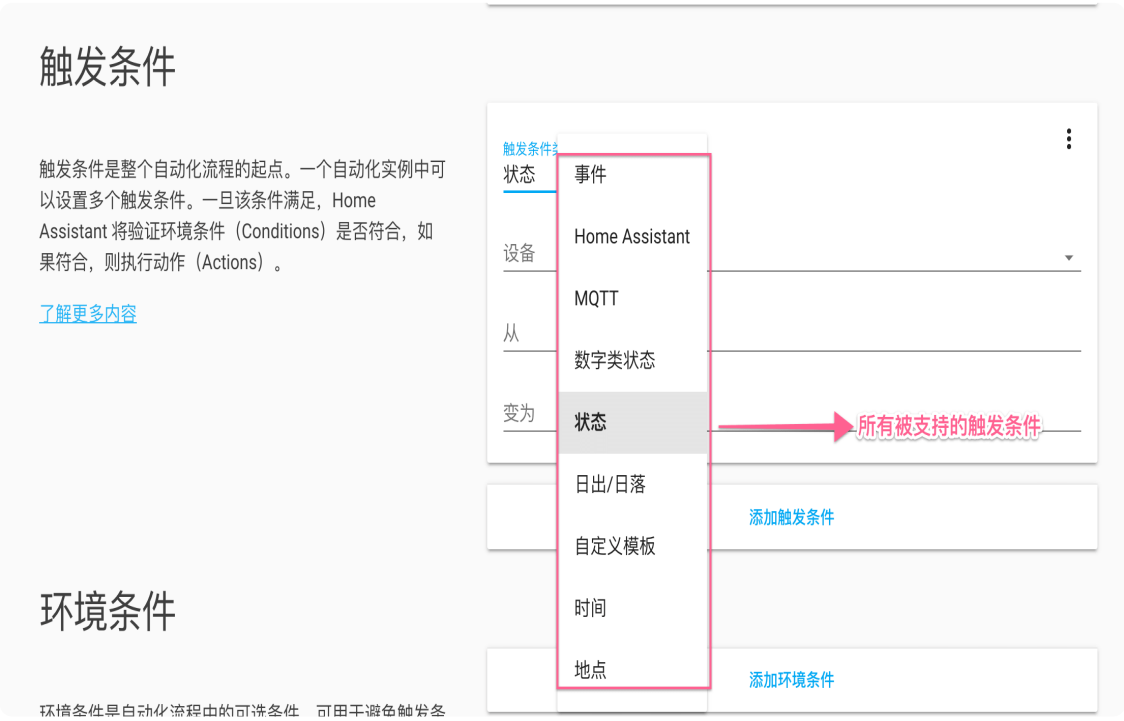


2. 点击页面右下角「加号」进入创建自动化任务页面，依次填入「触发条件」、「环境条件」、「动作」等内容，有没有觉得熟悉，对了它非常像 IFTTT 的创建步骤。



3. 触发条件设置：

先看一下 Home Assistant 的触发条件如何配置的：



我们看到它支持事件、状态、日出/日落、时间、地点等多种类型的触发条件，我们要监控的事「米家无线开关」被按住，很显然这是个事件，我们选择「事件」然后进行下一步：

接下来我们需要输入「事件类型」和「事件数据」，我们从哪里获取这些数据呢?

教给大家一个简单的办法，由于 Home Assistant 是命令行工具，所以它启动以后会在命令行中输出日志，我们打开命令行工具，然后按住「米家无线开关」，观察命令行输出的数据，会看到如下数据：

```
INFO:homeassistant.core:Bus:Handling <Event click[L]: entity_id=binary_sensor.switch_158d0001b7ebbf, click_type=long_click_press>
```

分析一下这个数据不难得出，「事件类型」= `click`，「事件数据」= `entity_id=binary_sensor.switch_158d0001b7ebbf, click_type=long_click_press`

由于「事件数据」要求是 Json 格式，所以事件数据如下：

```
{"click_type": "long_click_press","entity_id":"binary_sensor.switch_158d0001b7ebbf"}
```

所以「触发条件」最终配置如下：

4. 动作设置：当条件达成时 Home Assistant 也会执行对应的动作，这点很像 IFTTT，我们的动作是调用 IFTTT webhooks，我们先假设 webhooks 的名称为 `find_phone`，所以配置动作如图：

点击上图「保存」按钮后，返回上一步，可以看到创建好的任务已经在列表里了：

Tips：如果你的自动化任务并没有出现在这里，可以查看配置「自动化脚本」文档中是否创建成功了，命令行输入如下：
`vi ~/.homeassistant/automations.yaml`
如果能看到自己创建的内容就代表成功了。

第二步， 创建 IFTTT 的 webhooks 让手机发出警报

webhooks 我们已经比较熟悉了，接下来的问题就是如何让手机发出警报了。解决这个问题需要我们的一个老朋友 Pushover ，它在《第四章 | IFTTT 的进阶使用》和《番外篇 | IFTTT Notifications vs. Pushover》中都介绍过。

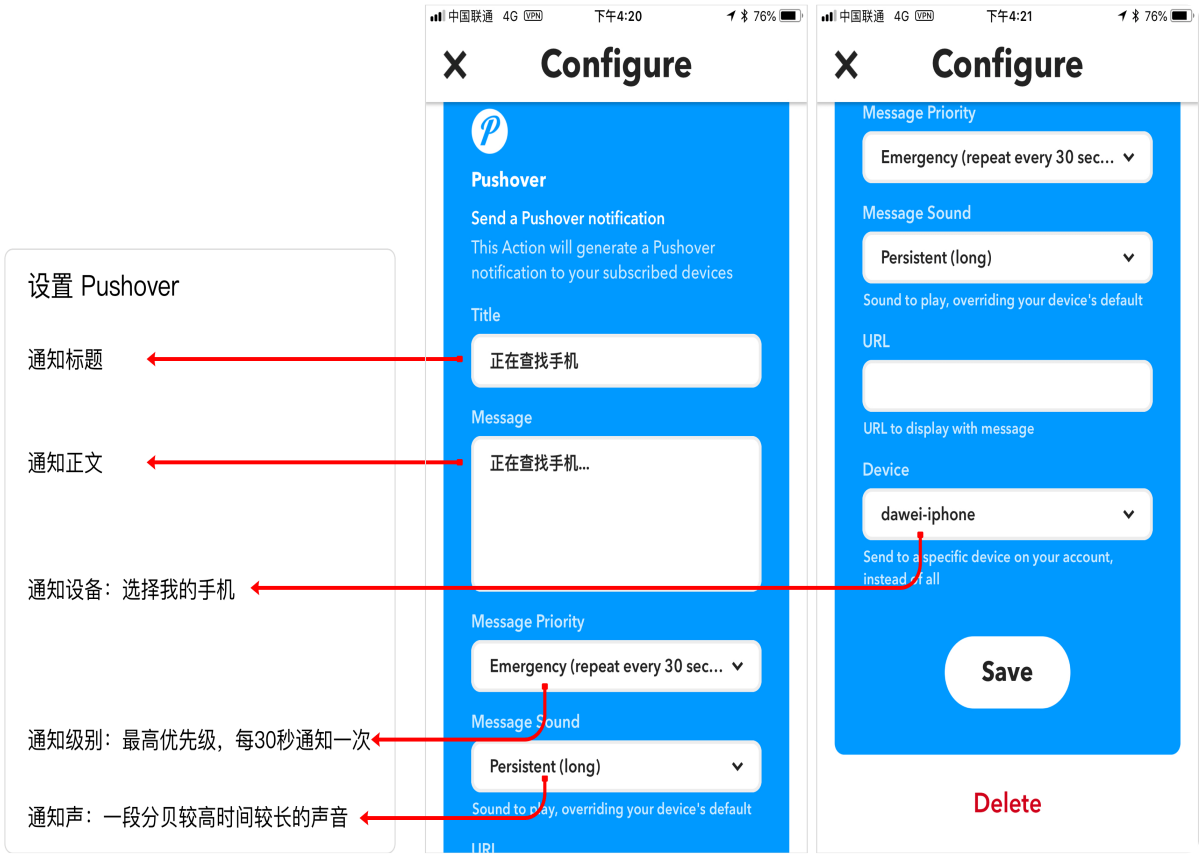
Pushover 把通知分成了五级：

1. **Lowest 最低优先级**：不会收到通知提醒，只在 App 中显示通知内容。
2. **Low 低优先级**：会收到通知提醒，但是没有声音或震动。
3. **Normal 标准优先级**：会收到通知提醒，也会有声音或震动。
4. **High 高优先级**：会收到通知提醒，也会有声音或震动，同时在通知内容在 App 内高亮显示。
5. **Emergency 最高优先级**：会收到通知提醒，每 30 秒重复提醒一次，直到确认为止。

在本事例中，我们希望手机能够发出警报声，最好能够持续的发出，这样我们才能方便地找到手机，很显然我们可以使用「Emergency 最高优先级」，它不

但持续地发出提醒，而且必须我们手动确认才能关闭提醒。

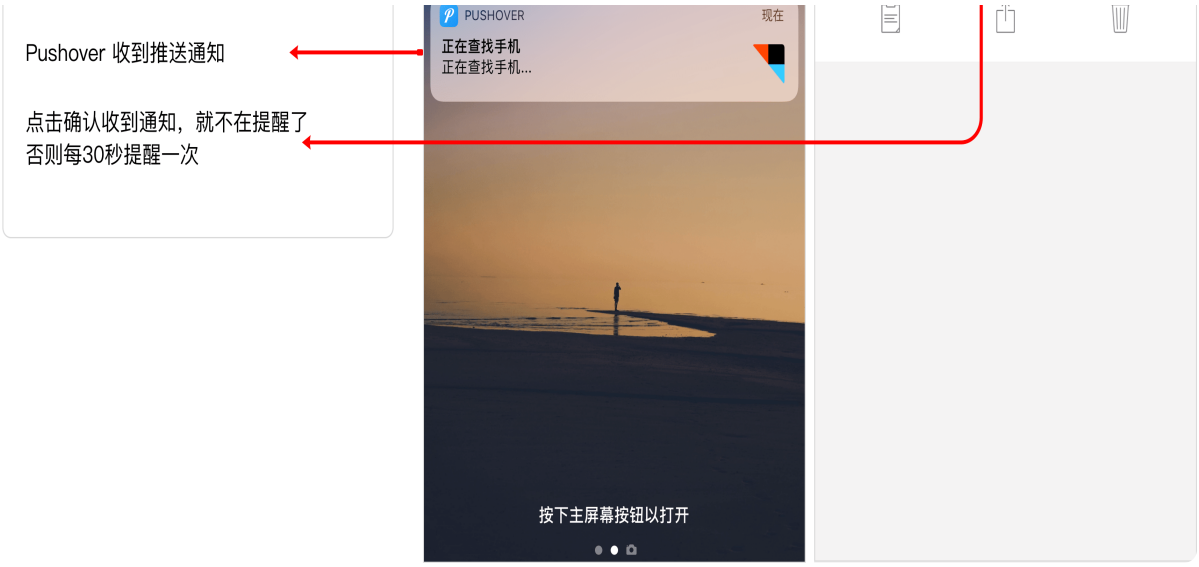
这一步我们应该很熟悉了，关键步骤如下：



设置完成后，只要名叫 **find_phone** 的 webhooks 被触发，手机就会每 30 秒发出一次高频警报声，直到我们手动确认关闭警报。

第三步，按住「米家无线开关」触发警报

当「米家无线开关」被按住，很快我们就收到 IFTTT 的推送了通知，如图：



小结

在事例 1 中，我们首先通过 Home Assistant 创建了自动化任务：

1. 触发条件：当「米家无线开关」被按住时。
2. 动作设置：通过 IFTTT 组件调用 webhooks ， 发送 Pushover 最高级别通知。

这个过程和我们之前创建 IFTTT 自动化任务非常像，相信大家会比较熟悉，接下来我们看下一个事例。

事例 2：如果 Gmail 收到重要的邮件，那么把米家网关灯设置为红色

分析一下这个需求：

1. Trigger ： 如果 Gmail 收到重要邮件。
2. Action：米家网关的灯设置为红色。

IFTTT 通过 **Gmail** 服务可以检测到「收到指定联系人的邮件」条件，这对于我们来说已经很简单了，下一步就是如何让 IFTTT 通知「米家网关」让它的灯变红色。

Home Assistant API

如果一个服务想要通过 IFTTT 自动化，必须提供 API，但是「米家」并没有提供 API，所以这里还要借助 Home Assistant 平台，Home Assistant 提供了友好的 API，供我们使用，如果想启用 API，需要设置访问密码以及白名单，否则会提示没有权限，操作如下：

1. 在命令行下输入以下命令打开配置文件 `vi ~/.homeassistant/configuration.yaml`。
2. 找到 `api_password` 这一行，去掉 `api_password` 前边的 `#`，并且将后边内容改为自己需要的密码。
3. 找到 `trusted_networks` 这一行，去掉前边的 `#`，并且追加白名单。

如图：

```
1  http:
2    # Secrets are defined in the file secrets.yaml
3    api_password: 12171217   设置密码
4    # Uncomment this if you are using SSL/TLS, running in Docker container, etc.
5    base_url: 127.0.0.1:8123 设置默认url
6    trusted_networks:
7      - 127.0.0.1    设置访问白名单
```

Tips：如果每次 Home Assistant 重启后打开的浏览器 IP 地址不是 `127.0.0.1`，可以将上图中的 `base_url` 内容改为 `127.0.0.1:8123`。

操作方法看视频：

主要操作指令（注意输入法要一直在英文状态下）：

1. 光标默认停在第一行第一个字符上。
2. 按方向键 `↓`，直到光标移动到 `#api_password:` 所在的行。
3. 此时光标在当前行第一个字符，按下 `I` 进入编辑状态。
4. 删除掉 `#api_password:` 前边的 `#`，并且将 `:` 后的 `password` 改成自己想要的密码 。
5. 密码改完后，按下 `Esc` 键，退出编辑状态。
6. 再按「下方向键」，直到光标移动到 `#trusted_networks:` 。
7. 此时光标在当前行第一个字符，按下 `I` 进入编辑状态。
8. 删除掉 `#trusted_networks:` 前边的 `#` 。
9. 按下 `Esc` 键，退出编辑状态。

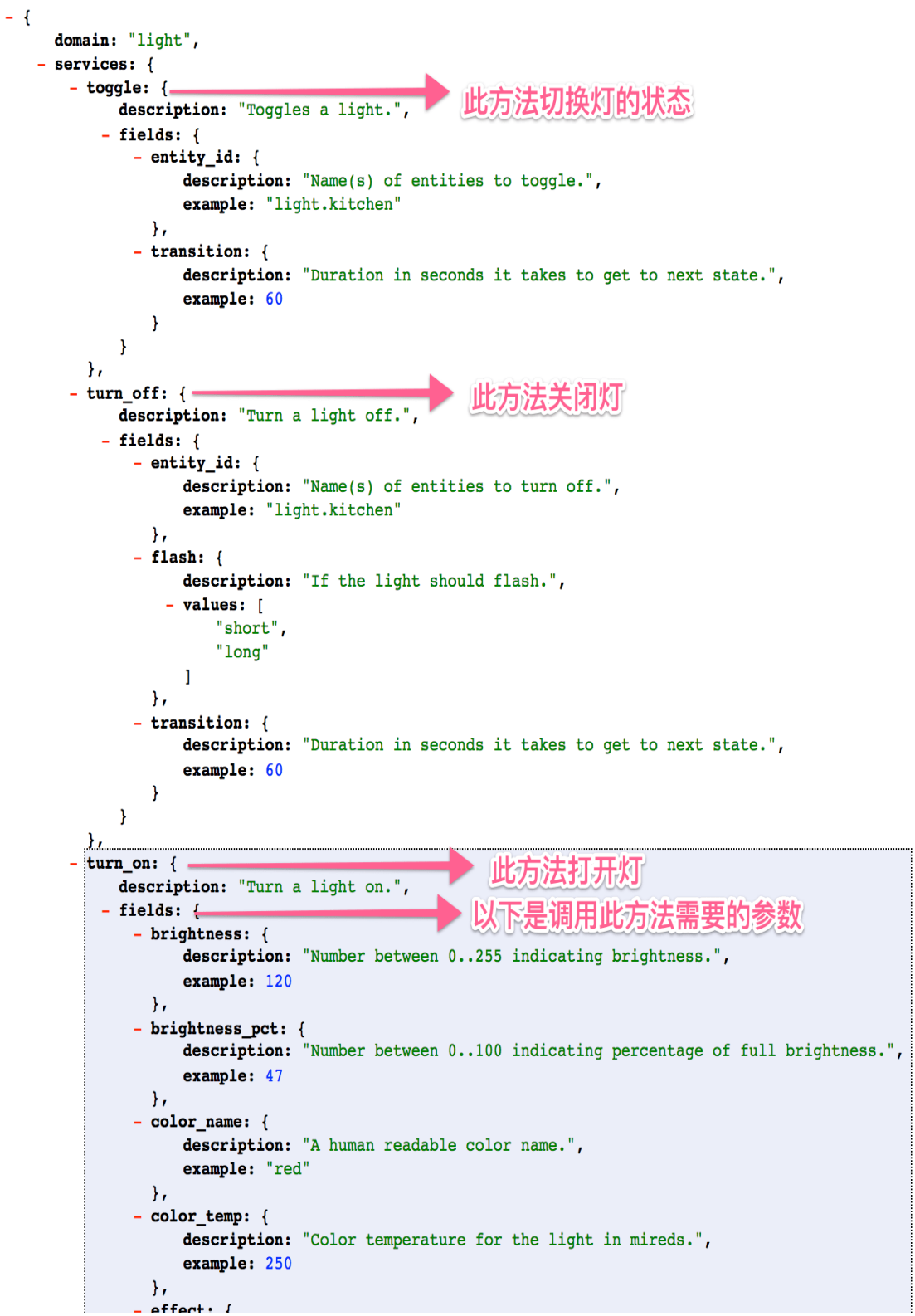
10. 再按方向键 `↓`，将光标移动到 `#trusted_networks:` 的下一行。
11. 此时光标在当前行第一个字符，按下 `i` 进入编辑状态。
12. 输入本机的 IP `127.0.0.1`。
13. 按下 `Esc` 键，退出编辑状态。
14. 按下 `:`，进入指令输入模式。
15. 在 `:` 后输入小写字母 `w` 后，配置文件修改被保存。
16. 最后再次按下 `:` 输入小写字母 `q`，退出编辑器，文件修改成功。

此时我们开启了 API，同时设置了访问密码，再次重启 Home Assistant，重启后，在浏览器中访问 API 页面：`http://127.0.0.1:8123/api/services`。

浏览器中会显示所有开放的 API 内容，其格式为 Json：


```
[
  - {
      domain: "homeassistant",
    - services: {
        - check_config: {
            description: "Check the Home Assistant configuration files for errors. Errors will be displayed in the Home Assistant log.",
            fields: { }
          },
        - reload_core_config: {
            description: "Reload the core configuration.",
            fields: { }
          },
        - restart: {
            description: "Restart the Home Assistant service. It is normal to get a "Failed to call service homeassistant/restart" message.",
            fields: { }
          },
        - stop: {
            description: "Stop the Home Assistant service. It is normal to get a "Failed to call service homeassistant/stop" message.",
            fields: { }
          },
        - toggle: {
            description: "Generic service to toggle devices on/off under any domain. Same usage as the light.turn_on, switch.turn_on, etc. services.",
          - fields: {
            - entity_id: {
                description: "The entity_id of the device to toggle on/off.",
                example: "light.living_room"
              }
            }
          },
        - turn_off: {
            description: "Generic service to turn devices off under any domain. Same usage as the light.turn_on, switch.turn_on, etc. services.",
          - fields: {
            - entity_id: {
                description: "The entity_id of the device to turn off.",
                example: "light.living_room"
              }
            }
          },
        - turn_on: {
            description: "Generic service to turn devices on under any domain. Same usage as the light.turn_on, switch.turn_on, etc. services.",
          - fields: {
```

这里列出来的是所有 Home Assistant 开放的 API，回到我们的需求，我们希望让「米家网关」亮红灯，所以我们在内容中找到打开灯的 API 如下图：




```
- {
    domain: "light",
  - services: {
    - toggle: {
        description: "Toggles a light.",
      - fields: {
        - entity_id: {
            description: "Name(s) of entities to toggle.",
            example: "light.kitchen"
          },
        - transition: {
            description: "Duration in seconds it takes to get to next state.",
            example: 60
          }
        }
      },
    - turn_off: {
        description: "Turn a light off.",
      - fields: {
        - entity_id: {
            description: "Name(s) of entities to turn off.",
            example: "light.kitchen"
          },
        - flash: {
            description: "If the light should flash.",
          - values: [
              "short",
              "long"
            ]
          },
        - transition: {
            description: "Duration in seconds it takes to get to next state.",
            example: 60
          }
        }
      },
    - turn_on: {
        description: "Turn a light on.",
      - fields: {
        - brightness: {
            description: "Number between 0..255 indicating brightness.",
            example: 120
          },
        - brightness_pct: {
            description: "Number between 0..100 indicating percentage of full brightness.",
            example: 47
          },
        - color_name: {
            description: "A human readable color name.",
            example: "red"
          },
        - color_temp: {
            description: "Color temperature for the light in mireds.",
            example: 250
          },
        - effect: {
```

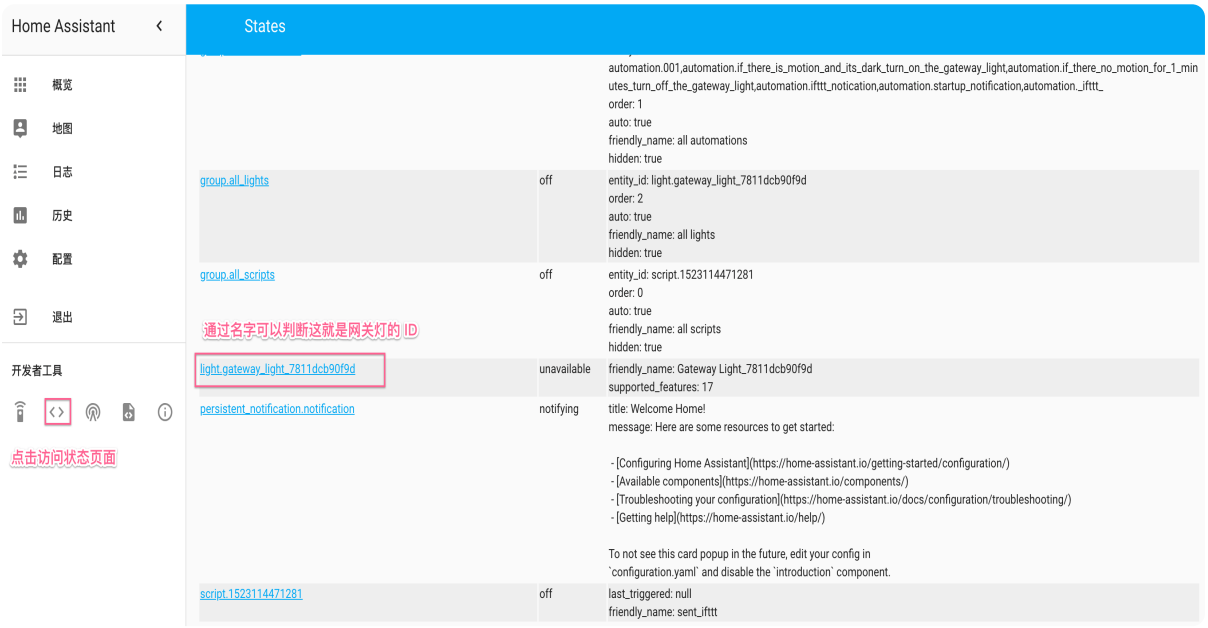
```
    description: "Light effect."
```

调用 API 使「米家网关」亮红灯

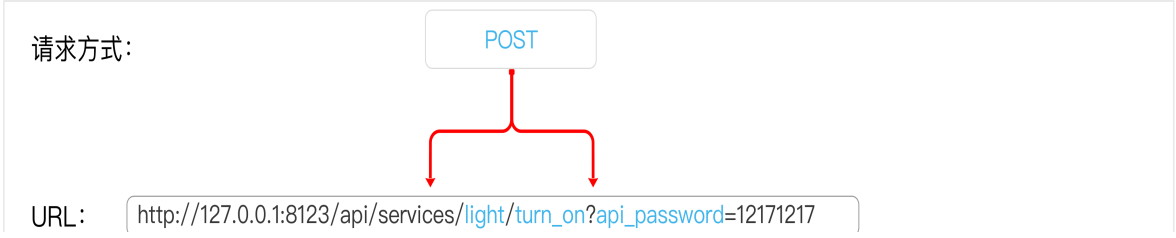
通过分析上图字段，可以很明显的发现，调用 `light` 服务的 `turn_on` 方法可以「打开灯」，其中参数意义如下：

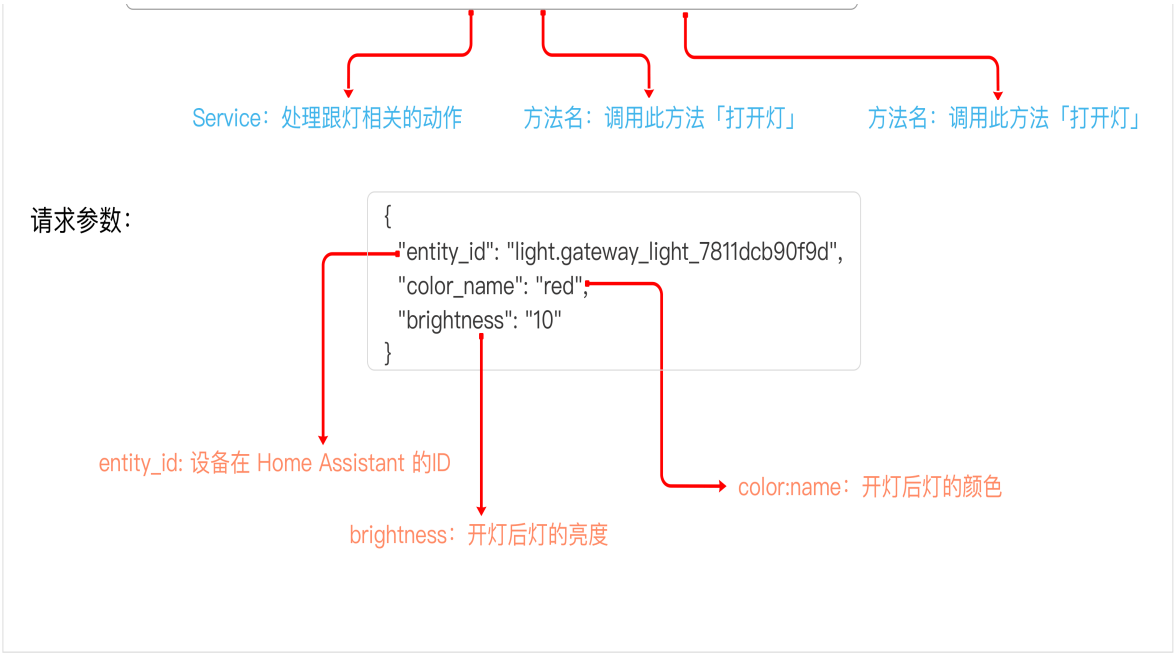
- `entity_id`：代表要开灯的智能硬件在 Home Assistant 中的 ID。
- `color_name`：设置灯的颜色，比如红色 *red*。
- `brightness`：设置灯的亮度，需要填写整数。

我们可以通过如下方式找到 `entity_id`：

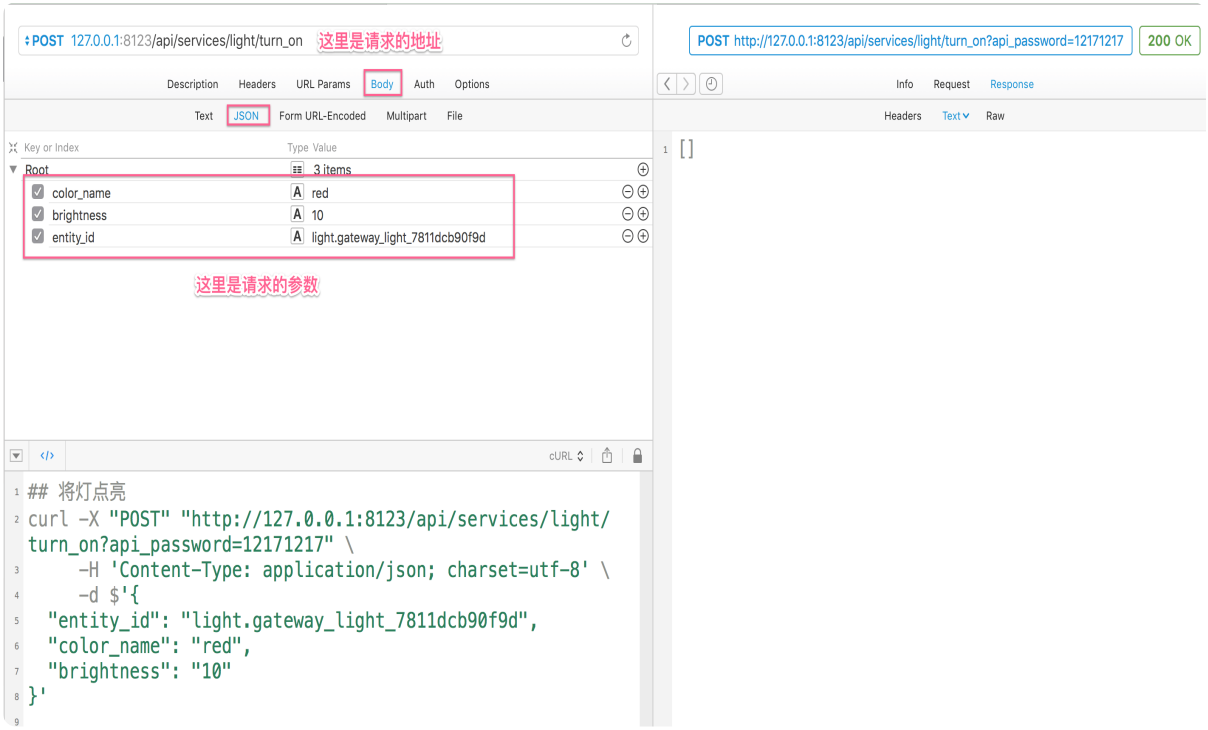


接下来我们需要通过 `POST` 方法请求 API，让灯亮红色，请求方法如图：

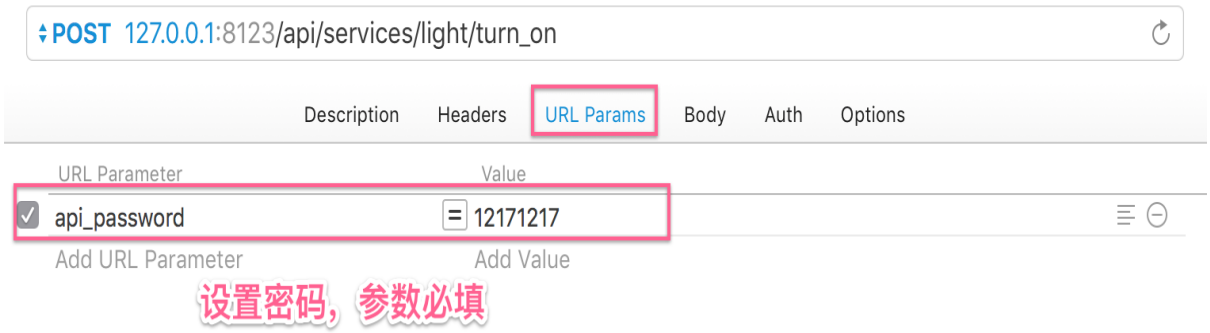




在 Mac OS 上我使用 Paw 软件进行 API 调用调试，其它平台推荐使用 Postman[4]，事例2 在 Paw 上的调用方式如图：



Tips：请求参数中一定要带着密码，参数设置如图：



执行 Paw 软件，发现网关灯成功被设置成红色，简单总结一下：

1. 通过 POST 访问如下地址

   ```
   http://127.0.0.1:8123/api/services/light/turn\_on?
   api\_password=12171217
   ```

   。

2. 传递如下参数：

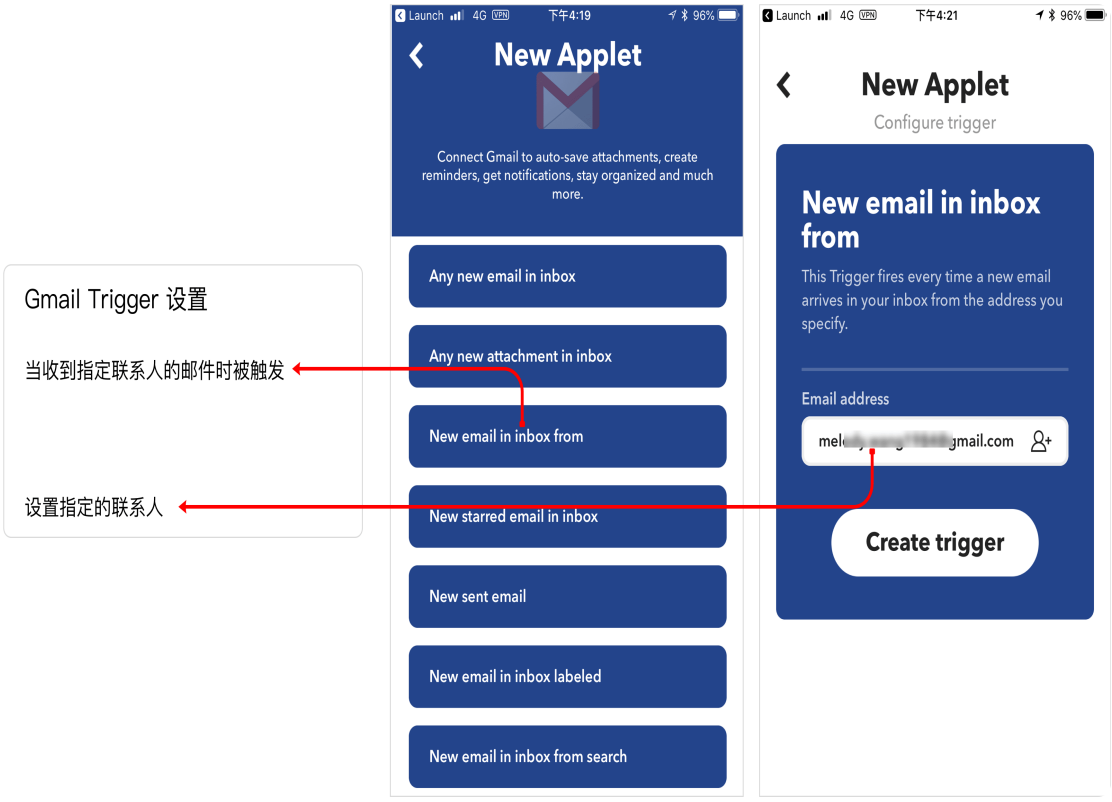
   ```
   "entity_id": "light.gateway_light_7811dcb90f9d",
   "color_name": "red",
   "brightness": "10"
   ```

既然有了方法，接下来我们通过 IFTTT 实现「如果 Gmail 收到重要的邮件，那么把米家网关灯设置为红色」。

IFTTT 中创建自动化任务

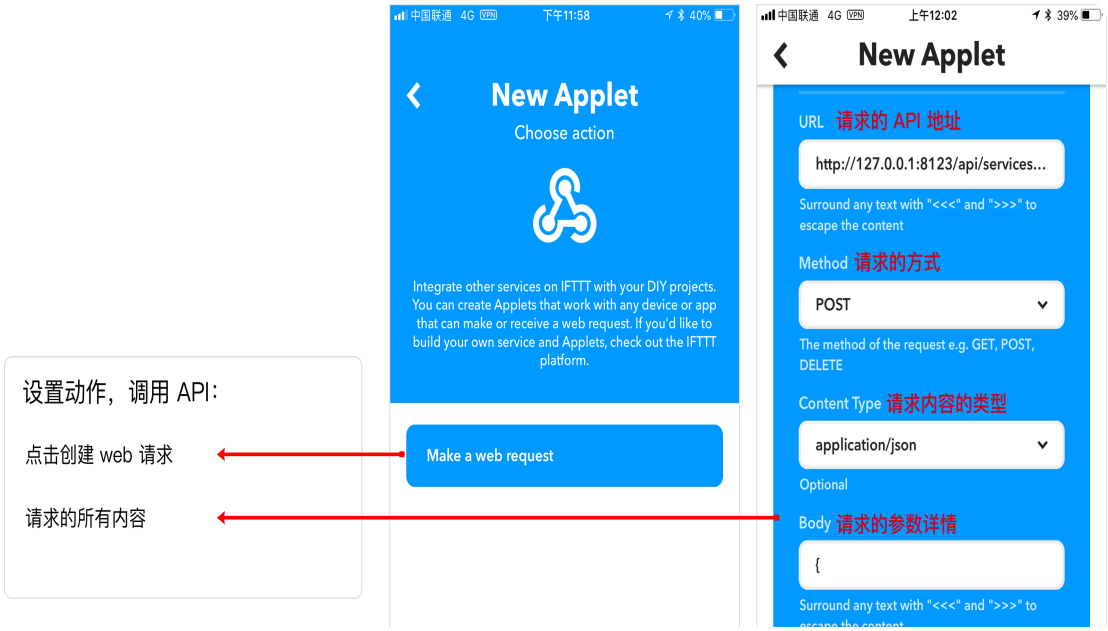
1. 设置 Trigger：

   Gmail 服务，设置条件为「如果收到指定联系人的邮件时」：


Gmail Trigger 设置
当收到指定联系人的邮件时被触发
设置指定的联系人
New Applet
Connect Gmail to auto-save attachments, create reminders, get notifications, stay organized and much more.
Any new email in inbox
Any new attachment in inbox
New email in inbox from
New starred email in inbox
New sent email
New email in inbox labeled
New email in inbox from search
New Applet
Configure trigger
New email in inbox from
This Trigger fires every time a new email arrives in your inbox from the address you specify.
Email address
Create trigger

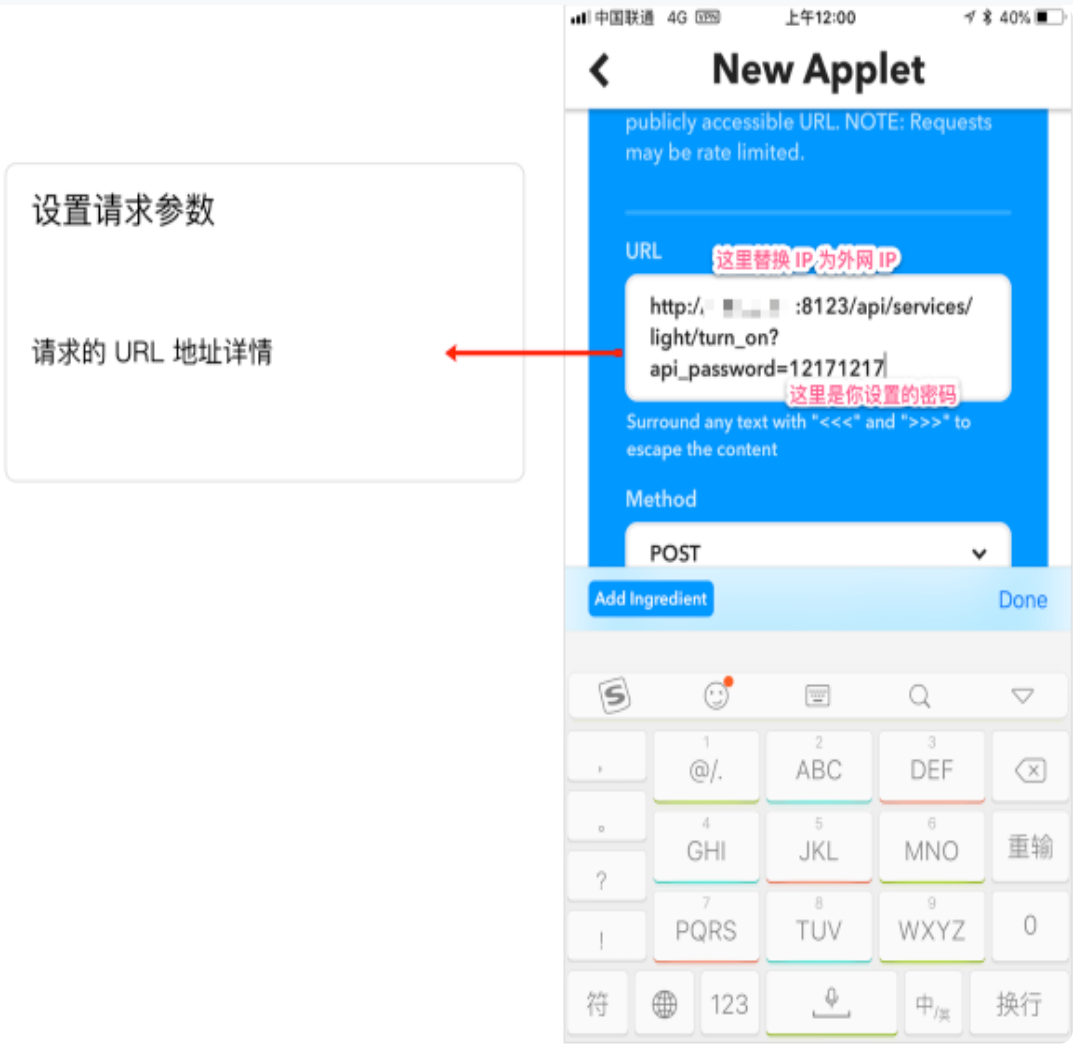
2. 设置 Action，POST 请求 Home Assistant 开灯的 API：

这里我们依然使用 **webhooks** 服务，当 **webhooks** 用在 IFTTT 的 Action 时，他的作用就是在 IFTTT 的服务端发起 Web 请求：


设置动作，调用 API：
点击创建 web 请求
请求的所有内容
New Applet
Choose action
Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.
Make a web request
New Applet
URL 请求的 API 地址
http://127.0.0.1:8123/api/services...
Surround any text with "<<<" and ">>>" to escape the content
Method 请求的方式
POST
The method of the request e.g. GET, POST, DELETE
Content Type 请求内容的类型
application/json
Optional
Body 请求的参数详情
{
Surround any text with "<<<" and ">>>" to

请求 URL 具体内容如下：

```
http://127.0.0.1:8123/api/services/light/turn_on?api_password=12171217
```

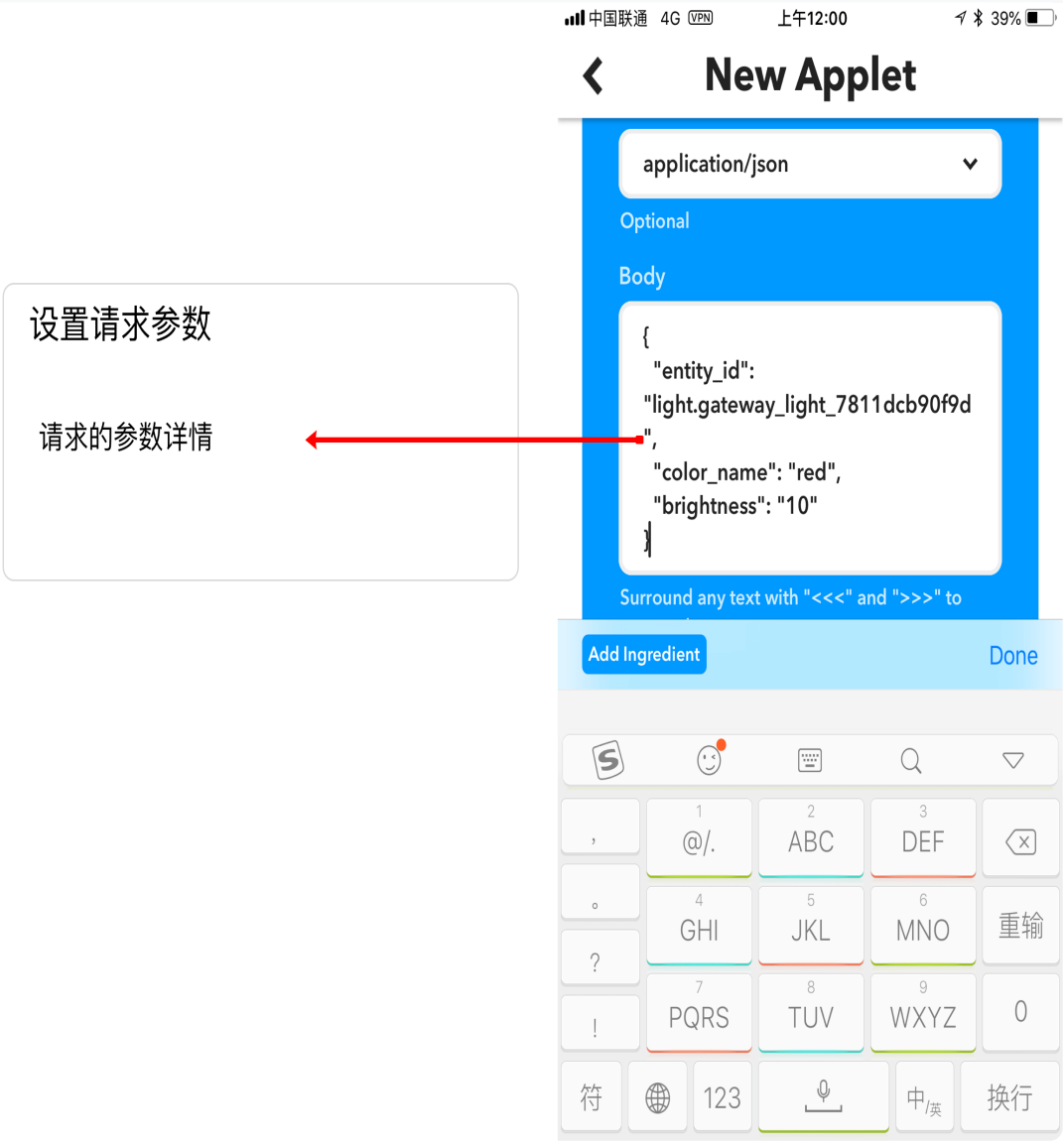


Tips： 大家注意这里的 IP 地址不能是 `127.0.0.1` 必须要换成公网 IP。通常我们家里的宽带都提供了动态的 IP（联系运营商可获得），在你的路由器上通过「动态解析域名」技术可以实现，具体方法请自行百度，这里

不做展开。（现在一般的路由器都内置了动态解析的组件，可以把内网的某一个机器的某一个端口映射到指定的域名端口。）

请求参数具体内容如下：

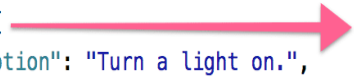
```
{"entity_id": "light.gateway_light_7811dcb90f9d","color_
name": "red","brightness": "10"}
```



小结

在事例 2 中，我们首先通过 Home Assistant 的 API 列表找到「打开灯」对应的的 API 函数如图：


```
{
    "domain": "light",
    "services": {
        "toggle": {
            "description": "Toggles a light.",
            "fields": {
                "entity_id": {
                    "description": "Name(s) of entities to toggle.",
                    "example": "light.kitchen"
                },
                "transition": {
                    "description": "Duration in seconds it takes to get to next state.",
                    "example": 60
                }
            }
        },
        "turn_off": {
            "description": "Turn a light off.",
            "fields": {
                "entity_id": {
                    "description": "Name(s) of entities to turn off.",
                    "example": "light.kitchen"
                },
                "flash": {
                    "description": "If the light should flash.",
                    "values": ["short", "long"]
                },
                "transition": {
                    "description": "Duration in seconds it takes to get to next state.",
                    "example": 60
                }
            }
        },
        "turn_on": {
            "description": "Turn a light on.",
            "fields": {
                "brightness": {
                    "description": "Number between 0..255 indicating brightness.",
                    "example": 120
                },
                "brightness_pct": {
                    "description": "Number between 0..100 indicating percentage of full
```



```
                "description": "Number between 0..100 indicating percentage of full
brightness.",
                "example": 47
            },
            "color_name": {
                "description": "A human readable color name.",
                "example": "red"
            },
            "color_temp": {
                "description": "Color temperature for the light in mireds.",
                "example": 250
            },
            "effect": {
                "description": "Light effect.",
                "values": ["colorloop", "random"]
            },
            "entity_id": {
                "description": "Name(s) of entities to turn on",
                "example": "light.kitchen"
            },
            "flash": {
                "description": "If the light should flash.",
                "values": ["short", "long"]
            },
            "kelvin": {
                "description": "Color temperature for the light in Kelvin.",
                "example": 4000
            },
            "profile": {
                "description": "Name of a light profile to use.",
                "example": "relax"
            },
            "rgb_color": {
                "description": "Color for the light in RGB-format.",
                "example": "[255, 100, 100]"
            },
            "transition": {
                "description": "Duration in seconds it takes to get to next state",
                "example": 60
            },
            "white_value": {
                "description": "Number between 0..255 indicating level of white.",
                "example": "250"
            },
            "xy_color": {
                "description": "Color for the light in XY-format.",
```

```
87                "example": "[0.52, 0.43]"
88            }
89        }
90    }
91  }
92 }
```

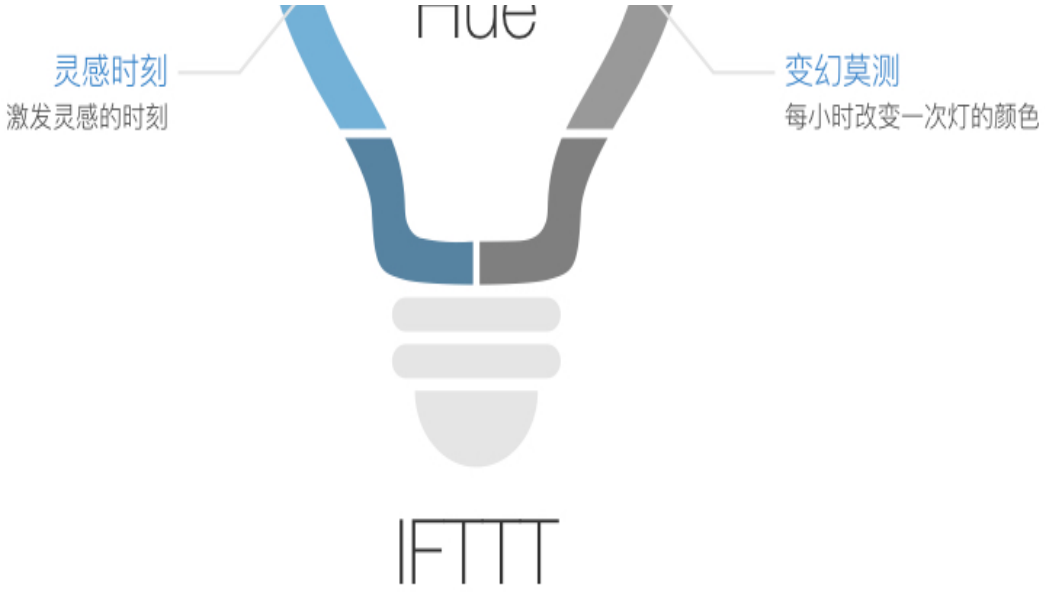
通过研究函数，我们发现请求 `light` 函数的 `turn_on` 方法，可以点亮指定的灯，URL和参数如下：

- URL：
  `http://127.0.0.1:8123/api/services/light/turn_on?api_password=12171217`
- 参数：

```
"entity_id": "light.gateway_light_7811dcb90f9d",
"color_name": "red",
"brightness": "10"
```

最后我们通过熟悉的方式，在 IFTTT 中创建「自动化任务」实现了「如果 Gmail 收到重要的邮件，那么把米家网关灯设置为红色」。

02. IFTTT 和 Philips Hue 的进阶玩法



Philips Hue，Philips 出品的智能灯控系统，它包括一个控制器和若干的灯具，灯具和控制器之间属于物联网范畴；同时控制器和互联网相连，通过网络我们可以给控制器发送指令，从而控制灯具。最新一代 Philips 控制器符合 Apple HomKit 的标准，也就是我们可以通过 Siri 控制开灯关灯。

接下来我们体验 Philips Hue 玩法，由于 Philips Hue 自身提供了开放的 API，所以我们无需接入 Home Assistant 平台，直接在 IFTTT 中设置即可。

Tips：如果跟着步骤操作，需要「Hue 桥接器」和至少一个 Hue 智能灯，这里推荐「Hue Go 智能魔灯」。

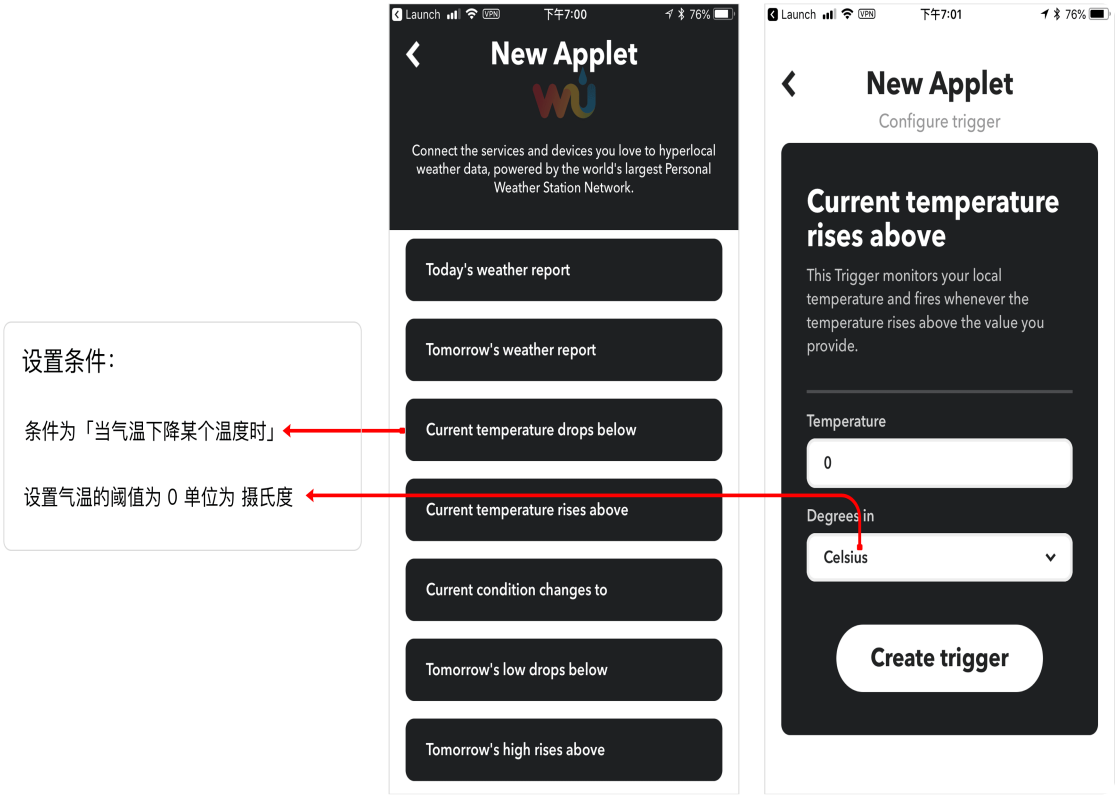
事例 3：当气温低于 0 度，Philips Hue 的颜色改为冷色调

照例先我们分析一下需求：

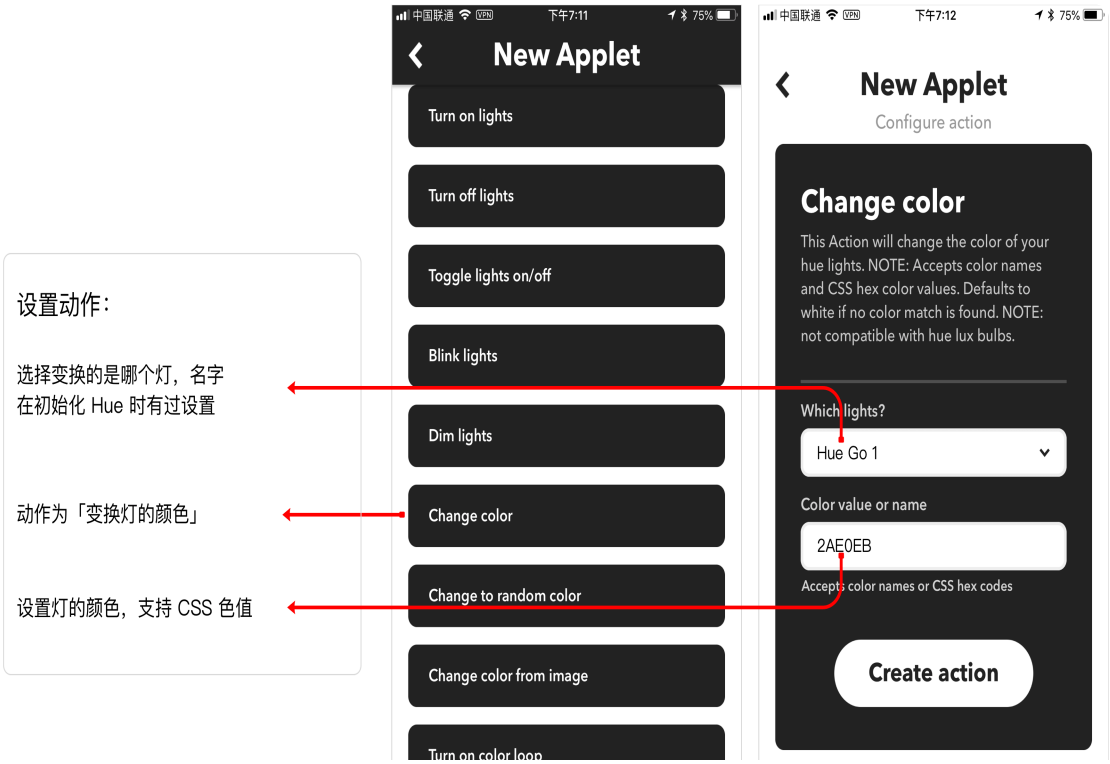
- 触发器：当前气温低于 0 摄氏度
- 动作：把 Philips Hue 的灯颜色调整为冷色调

我们依然通过 **Weather Underground** 服务获取天气信息。

1. 创建 Trigger，设置条件为气温低于零摄氏度时：



2. 创建 Action，将 Philips Hue 的颜色改为冷色调：



事例 4：在指定的时间点，将 Hue 自动关闭

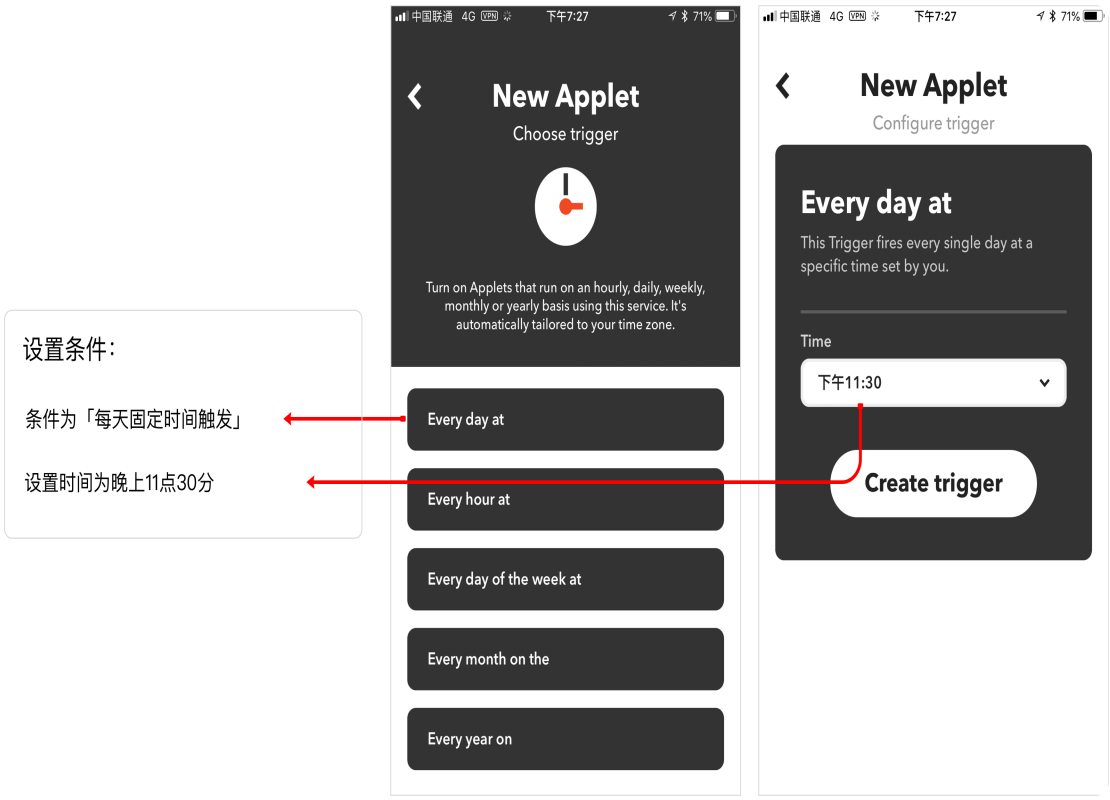
有时我们会在睡觉前忘记了关掉部分灯，比如廊灯，比如玄关灯，利用 IFTTT 我们可以简单的设置一个时间，到了那个时间，所有的灯都会被关闭。

先分析一下需求

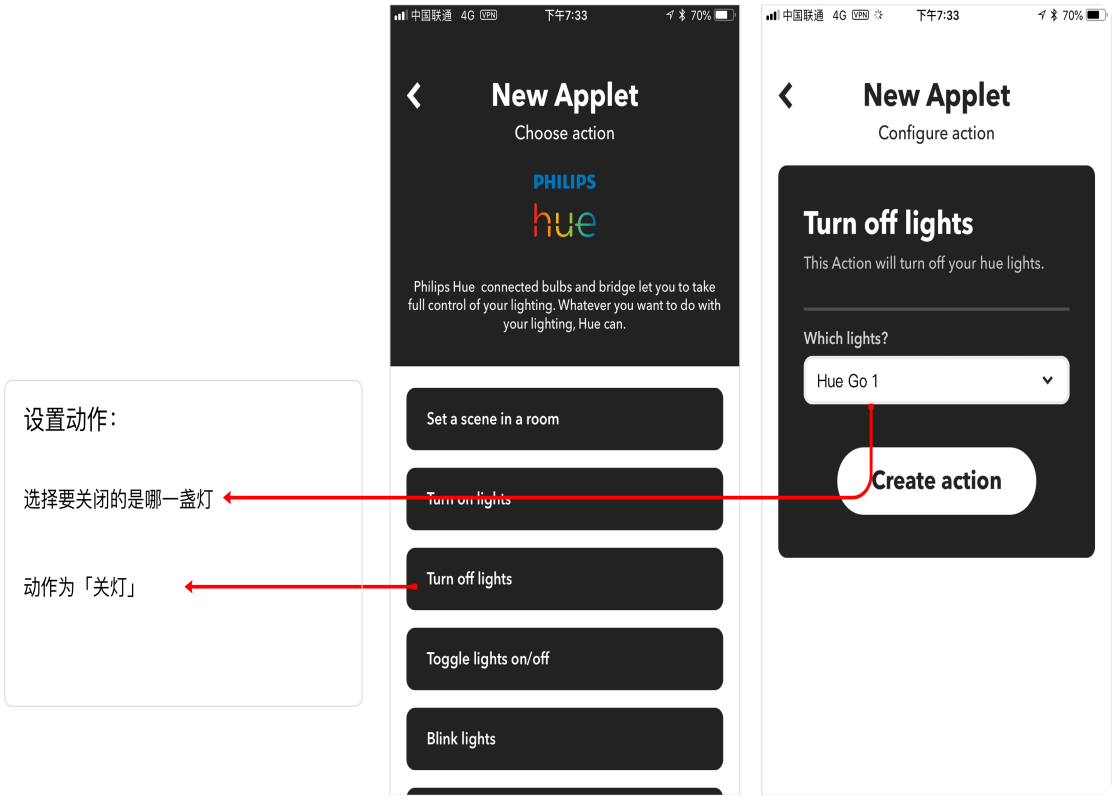
1. 设置一个具体的时间点：每天晚上 23:30
2. 如果时间到了，关掉指定的灯（走廊灯，玄关灯）

我们使用 **Date & Time** 服务制作定时任务：

1. 创建 Trigger，条件为每天晚上 23:30：



2. 创建 Action，动作为关闭 Hue 灯：



事例 5：设置起床时间，准时用灯光唤醒

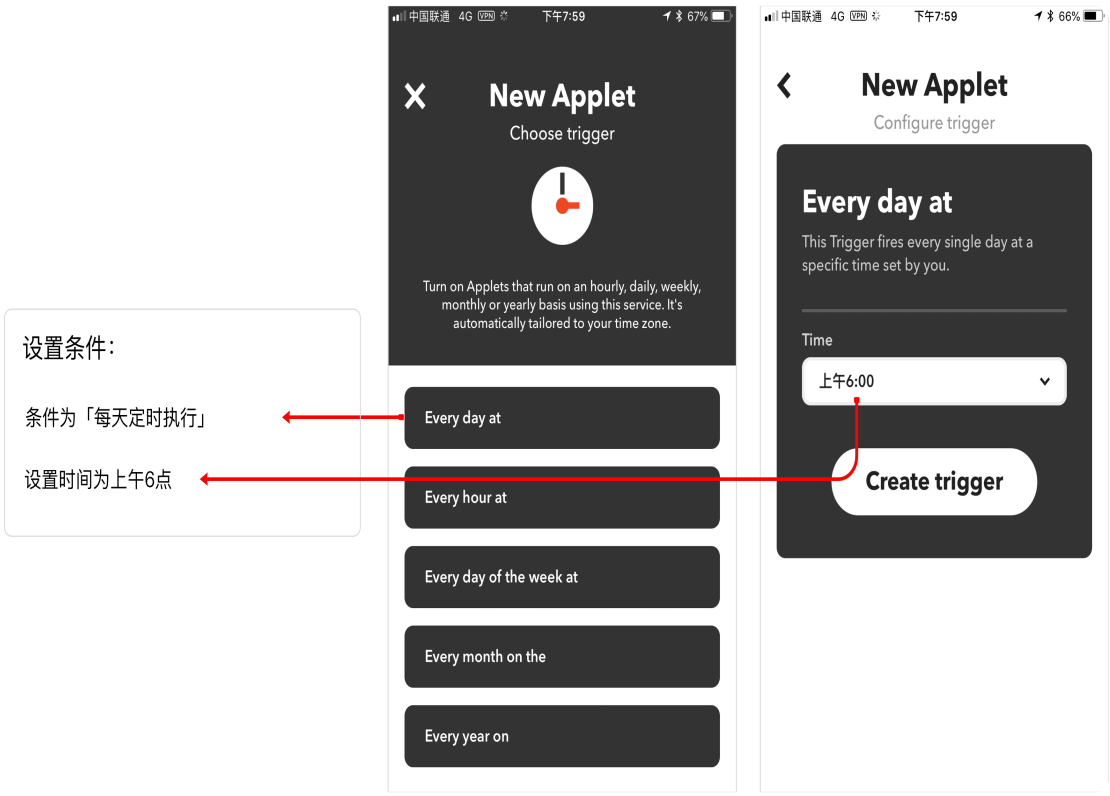
设置一个特定时间开灯，比如我们早上闹钟响了，让灯亮起来，这样有助于我们起床，通过 IFTTT 可以很方便的实现此功能。

先分析具体需求：

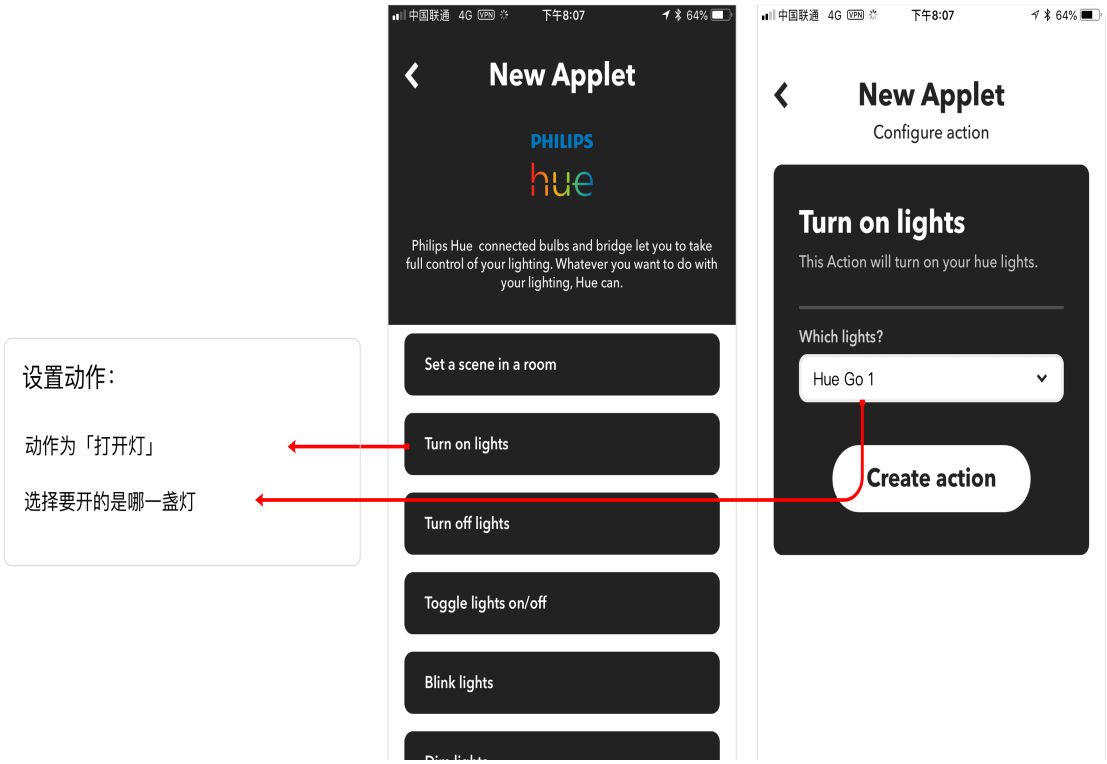
1. 设置好我们的起床时间
2. 时间到了的时候，点亮 Hue

我们继续使用 **Date & Time** 服务制作定时任务：

1. 创建 Trigger，条件为每天早上 6 点：


设置条件：
条件为「每天定时执行」
设置时间为上午6点
New Applet
Choose trigger
Turn on Applets that run on an hourly, daily, weekly, monthly or yearly basis using this service. It's automatically tailored to your time zone.
Every day at
Every hour at
Every day of the week at
Every month on the
Every year on
New Applet
Configure trigger
Every day at
This Trigger fires every single day at a specific time set by you.
Time
上午6:00
Create trigger

2. 创建 Action，打开 Hue 灯：



事例 6：每小时改变一次灯的颜色

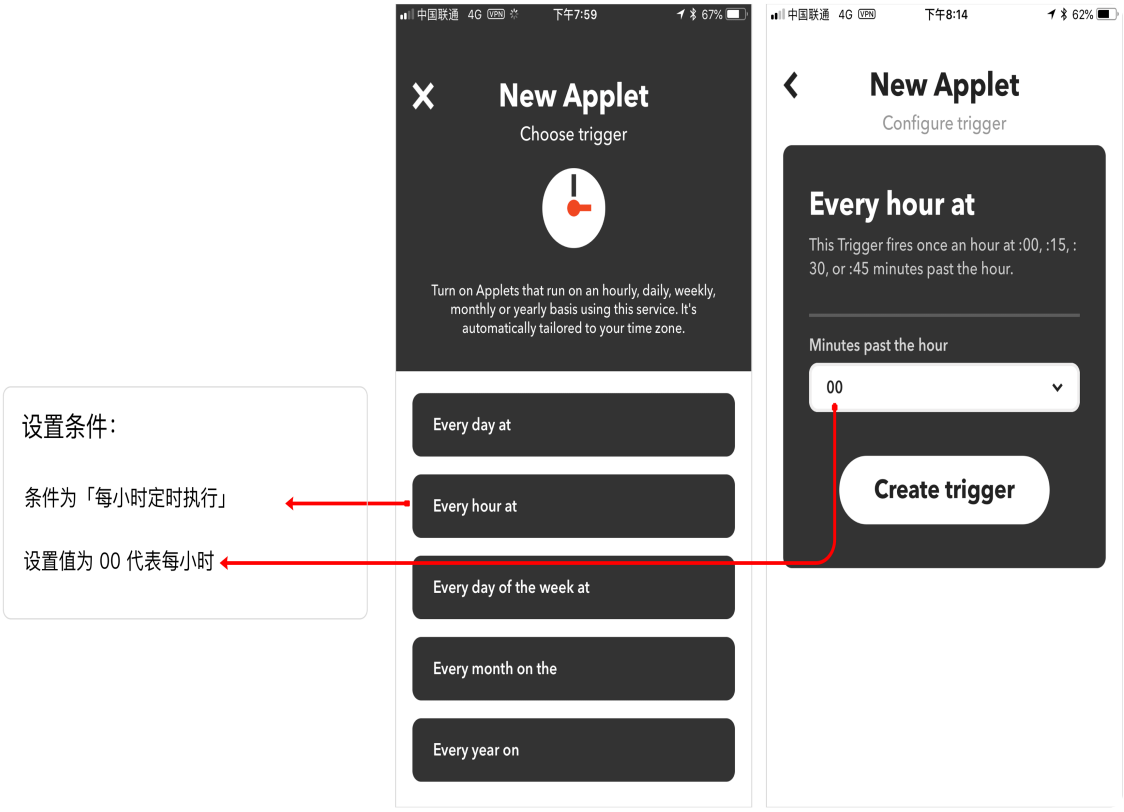
如果你正在组织家庭聚会，为了让氛围更活泼，可以设置每小时改变一下灯的颜色，这样的氛围相信小伙伴们一定会喜欢的。

先分析具体需求：

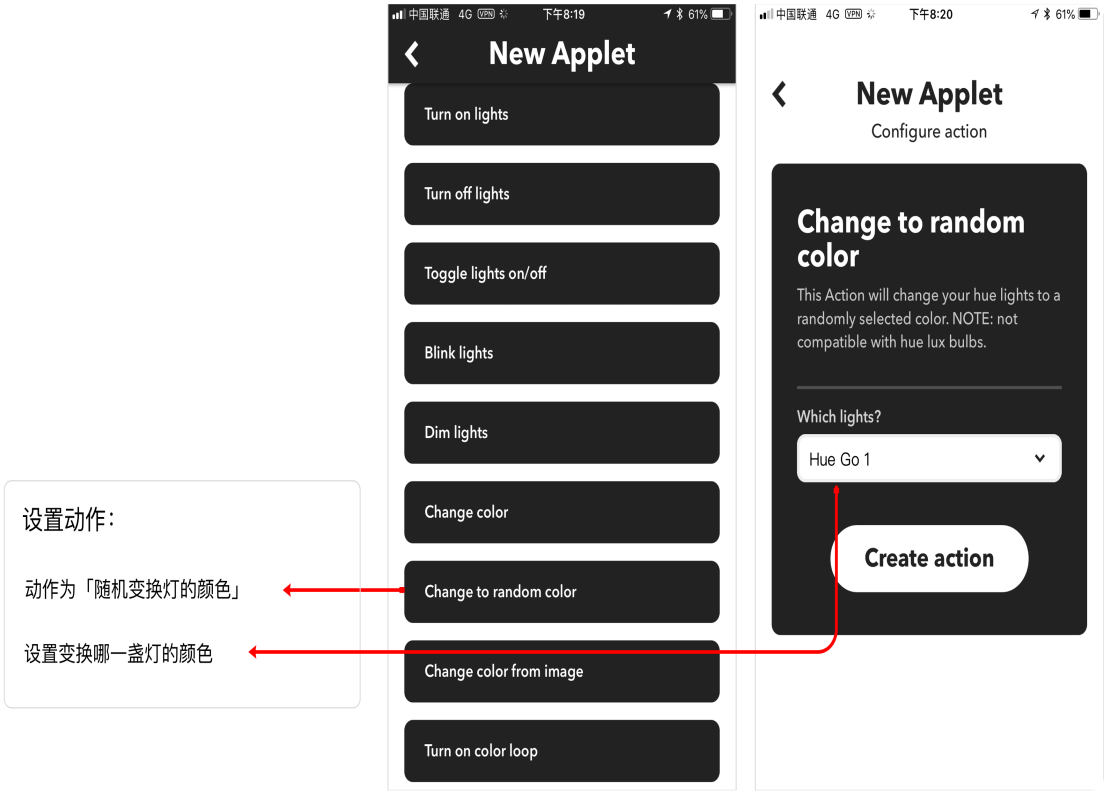
1. 设置定时器，每小时执行一次
2. IFTTT 改变 Hue 灯的颜色

使用 **Date & Time** 服务制作定时任务：

1. 创建 Trigger，条件为每天小时执行一次：



2. 创建 Action，动作为随机变换等的颜色：



04. 本篇总结

本篇介绍了两个智能硬件平台「米家」和「Philips Hue」，以及它们结合 IFTTT 的一些进阶玩法，涉及到了大量的命令行操作，希望大家不要产生惧怕，其实本篇案例中的命令行操作只涉及到移动、编辑、保存、退出，只要稍加阅读就能掌握，况且如果你想要进行更多进阶的玩法，掌握 VI 会对你有很大的帮助。

这是本系列正文的最后一章，《玩转 IFTTT》系列即将更新完毕，本系列中通过大量的事例操作，带着大家熟悉并了解 IFTTT，相信大家跟着教程，一步步到今天一定掌握了大量的操作方法。「自动化」这件事本身不重要，最重要的是如何做自动化？即我们可以把哪些事情自动化？这是一个持续思考和实践的过程，教程虽然更新完了，但是 IFTTT 可以做的事情还有很多，大家可以根据

自己的实际案例和想法尝试各种不同的用法，真正的发挥自动化的效果，打造属于自己的智能生活。关于 IFTTT 使用中有什么问题，依然可以在本教程的文章中进行评论或者私信我提问，我依然会定时更新评论和答疑汇总。

最后我想说，技术在更新，环境在变化，我们追求效率生活的脚步是不能停止的，希望大家能够保持学习的心态，把效率当成一种生活态度。

---

1. 一般说的「命令行」是指 linux 命令，linux 命令是对 Linux 系统进行管理的命令。对于 Linux 系统来说，无论是中央处理器、内存、磁盘驱动器、键盘、鼠标，还是用户等都是文件，Linux 系统管理的命令是它正常运行的核心，与之前的 DOS 命令类似。linux 命令在系统中有两种类型：内置 Shell（外壳）命令和 Linux 命令。 ↩
2. vi 是 unix/linux 操作系统中最经典的文本编辑器，只能是编辑字符，不能对字体、段落进行排版;它既可以新建文件，也可以编辑文件；他没有菜单，只有命令。 ↩
3. Postman，强大的跨平台 API 调试工具，推荐使用，如果无法访问网站，可以 FQ 后再次尝试。 ↩

上一期
番外篇 | IFTTT Notifications vs. ...